THE UNCAPACITATED FACILITY LOCATION PROBLEM[+,++]

by

Gerard Cornuejols[*]
George L. Nemhauser[**]
and
Laurence A. Wolsey[***]

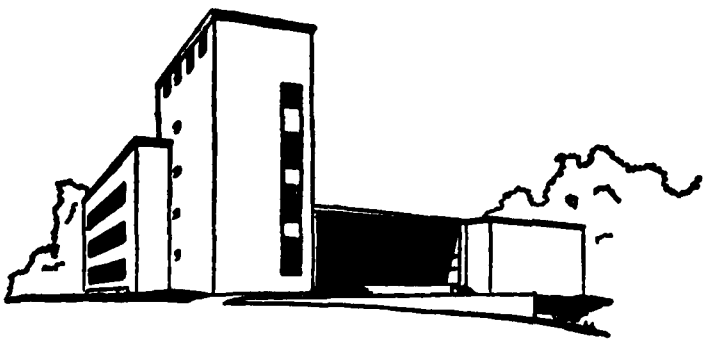
August. 1983

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



(1)

Management Science Research Report No. MSRR 493

THE UNCAPACITATED FACILITY LOCATION PROBLEM[+,++]

by

Gerard Cornuejols[*]
George L. Nemhauser[**]
and
Laurence A. Wolsey[***]

August, 1983

[+] This paper was prepared as a Chapter for the forthcoming book Discrete Location Theory, R.L. Francis and P. Mirchandini (eds.), Wiley-Interscience.

[++] This work was supported in part by National Science Foundation Grants ECS-8005350 and ECS-8205425.

[*] Graduate School of Industrial Administration, Carnegie-Mellon University

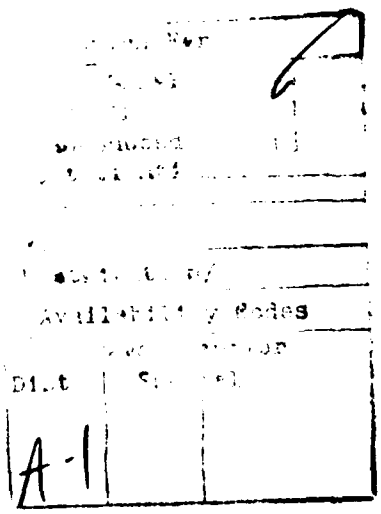
[**] School of Operations Research and Industrial Engineering, College of Engineering, Cornell University

[***] Center for Operations Research and Econometrics, Universite Catholique de Louvain

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract No. N00014-82-K-0329 NR 047-607 with the U. S. Office of Naval Research and in part by the National Science Foundation grants. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Table of Contents



1. Formulation and Applications

An economic problem of great practical importance is to choose the location of facilities, such as industrial plants or warehouses, in order to minimize the cost (or maximize the profit) of satisfying the demand for some commodity. In general there are fixed costs for locating the facilities and transportation costs for distributing the commodities between the facilities and the clients. This problem has been extensively studied in the literature and is often referred to as the plant, warehouse or facility location problem. When each potential facility has a capacity, which is the maximum demand that it can supply, the problem is known as the capacitated facility location problem. When the capacity hypothesis is not needed, we have the simple or uncapacitated facility location problem, which we abbreviate by UL.

The mathematical formulation of these problems as integer programs has proven very fruitful in the derivation of solution methods. To formalize UL, we consider a set of clients $I = \{1,\ldots,m\}$ with a given demand for a single commodity, and a set of sites $J = \{1,\ldots,n\}$ where facilities can be located. In the literature it has been traditional to use the phrasing "facility j is open" to mean that a facility is actually located at site j. Let f_j be the given fixed cost of opening facility j and assume there is a known profit c_{ij} that is made by satisfying the demand of client i from facility j. Typically, c_{ij} is a function of the production costs at facility j, the transportation costs from facility j to client i, the demand of client i and the selling price to client i. For example, $c_{ij} = d_i(p_i - q_j - t_{ij})$ where d_i is the demand, p_i the price per unit, q_j the production cost per unit and t_{ij} the transportation cost per unit. UL

is to open a subset of facilities in order to maximize total profit, given that all demand has to be satisfied.

For any given set S of open facilities, it is optimal to serve client i from a facility j for which c_{ij} is maximum over $j \in S$. So, given S, the profit is $z(S) = \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j$. The problem is to find a set S that yields the maximum profit Z, i.e. $Z = \max_{S \subseteq J} z(S)$. This can be viewed as a combinatorial formulation of the problem. Note that there is no loss of generality in assuming $f_j \geq 0$ for all $j \in J$ since if $f_k < 0$ every optimal solution contains facility k.

An integer linear programming formulation is obtained by introducing the following variables. Let $x_j = 1$ if facility j is open and $x_j = 0$ otherwise; $y_{ij} = 1$ if the demand of client i is satisfied from facility j and $y_{ij} = 0$ otherwise. The integer program is

$$(1.1) \qquad Z = \max \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j$$

$$(1.2) \qquad \sum_{j \in J} y_{ij} = 1 \qquad \text{all } i \in I$$

$$(1.3) \qquad y_{ij} \leq x_j \qquad \text{all } i \in I,\ j \in J$$

$$(1.4) \qquad x_j,\ y_{ij} \in \{0,1\} \quad \text{all } i \in I,\ j \in J.$$

The constraints (1.2) guarantee that the demand of every client is satisfied, whereas (1.3) guarantees that the clients are supplied only from open facilities.

The selling price of the commodity to client i enters the solution of (1.1)-(1.4) as a constant. Only the costs (production, transportation and fixed operating costs) are relevant for the decision. This is why in the literature UL is often presented as

$$\text{(1.1')} \qquad \min \sum_{i \in I} \sum_{j \in J} d_{ij} y_{ij} + \sum_{j \in J} f_j x_j$$

subject to (1.2)-(1.4) where d_{ij} is the cost (production plus transportation) of serving client i from facility j. This formulation is mathematically equivalent to (1.1)-(1.4) since (1.1') becomes -(1.1) by setting $c_{ij} = -d_{ij}$. In other words, costs can simply be regarded as negative profits.

In the integer program (1.1)-(1.4), the x_j's are the important variables. Once a set of x_j's that satisfy (1.4) are specified, it is simple to determine a set of y_{ij}'s that solves the integer program for the fixed x_j's. Even if we drop the integrality requirement on the y_{ij}'s, an optimal set of y_{ij}'s is given by $y_{ij^*} = 1$ and $y_{ij} = 0$, $j \neq j^*$ where $c_{ij^*} = \max\{c_{ij}: x_j = 1, j \in J\}$. Thus (1.4) can be replaced by

$$\text{(1.4')} \qquad x_j \in \{0,1\},\ y_{ij} \geq 0 \quad \text{all } i \in I,\ j \in J.$$

The formulation (1.1)-(1.3), (1.4') is a mixed integer linear program.

An integer linear program equivalent to (1.1)-(1.4) is obtained by replacing the constraints (1.3) by the more compact set of constraints

$$(1.3') \qquad \sum_{i \in I} y_{ij} \leq m x_j \quad \text{all} \quad j \in J.$$

To see that (1.3) can be replaced by (1.3'), note that when $x_j = 0$ both (1.3) and (1.3') imply $y_{ij} = 0$ for all $i \in I$, and when $x_j = 1$ both (1.3) and (1.3') are satisfied for all choices of y_{ij} that satisfy the constraints (1.2). However, the two formulations are equivalent only for 0-1 values of the variables x_j. For each j, (1.3') is obtained by summing (1.3) over all $i \in I$; hence any solution to (1.2), (1.3) is also a solution to (1.2) and (1.3'). But the converse is false when $0 < x_j < 1$; i.e. the feasible region defined by (1.2), (1.3') and

$$(1.5) \qquad 0 \leq x_j \leq 1, \; y_{ij} \geq 0 \quad \text{all} \quad i \in I, \; j \in J$$

strictly contains the region defined by (1.2), (1.3) and (1.5).

In UL, the number of facilities that are open in an optimal solution is not specified; it is determined by a solution of (1.1)-(1.4). From a practitioner's standpoint, it might be useful to consider a formulation where the number p of open facilities is a parameter of the problem. This is realized by adding one of the following constraints to the program (1.1)-(1.4)

(1.6) $$\sum_{j \in J} x_j = p$$

or

(1.7) $$\sum_{j \in J} x_j \leq p$$

where p is some given integer, $1 \leq p \leq n$. The formulation (1.1)-(1.4), (1.6) has been called the p-facility location problem. When $n = m$ and $f_j = 0$ for all $j \in J$, (1.1)-(1.4), (1.6) is known as the p-median problem. It is often solved in the literature using the minimization objective function (1.1') where (d_{ij}) is a distance matrix, i.e. it is positive, symmetric and $d_{ij} + d_{jk} \geq d_{ik}$, $1 \leq i,j,k \leq n$. This model is developed in Chapter 3 of the book.

To motivate further the study of UL, we give two examples. First we interpret (1.1)-(1.4) as a bank account location problem, see Cornuejols, Fisher, Nemhauser (1977b). The second example occurs in clustering analysis, see Mulvey and Crowder (1979). Other examples arise in lock-box location [Kraus, Janssen and McAdams (1970)] location of offshore drilling platforms [Balas and Padberg (1976)], economic lot sizing [Krarup and Bilde (1977)], machine scheduling and information retrieval [Hansen and Kaufman (1972)] and portfolio management [Beck and Mulvey (1982)].

A Bank Account Location Problem

The number of days required to clear a check drawn on a bank in city j depends on the city i in which the check is cashed. Thus, to maximize its available funds a company that pays bills to clients in various locations may find it advantageous to maintain accounts in several strategically located

banks. It would then pay bills to clients in city i from a bank in city j that had the largest clearing time. The economic significance to large corporations of such a strategy is discussed in an article in Businessweek (1974).

To formalize the problem of selecting an optimal set of account locations, let $I = \{1,\ldots,m\}$ be the set of client locations, $J = \{1,\ldots,n\}$ the set of potential account locations, f_j the fixed cost of maintaining an account in city j, d_i the dollar volume of checks paid in city i, and ϕ_{ij} the number of days (translated into monetary value) to clear a check issued in city j and cashed in city i. All this information is assumed to be known and $c_{ij} = d_i\phi_{ij}$ represents the value of paying clients in city i from an account in city j. Let $x_j = 1$ if an account is maintained in city j, and $x_j = 0$ otherwise; $y_{ij} = 1$ if clients in city i are paid from account j, and $y_{ij} = 0$ otherwise. Then the account location problem can be stated as (1.1)-(1.4).

Besides desiring to delay payments for as long as possible, corporations also want to collect funds due to them as quickly as possible. This can be done by situating check collection centers or "lock-boxes" at optimal locations. Since (1.1) and (1.1') are mathematically equivalent, (1.1)-(1.4) is also a model for the lock-box problem.

A Clustering Problem

Cluster analysis consists of partitioning objects into classes, known as clusters, in such a way that the elements within a cluster have a high degree of natural association among themselves while the clusters are relatively distinct from one another. Cluster analysis is used in biology, psychology,

medicine, artificial intelligence, pattern recognition, marketing research, automatic classification and statistics.

Let $I = \{1,\ldots,m\}$ be the set of objects to be clustered. The clustering is done around objects that "represent" the clusters. Let $J = \{1,\ldots,n\}$ be the set of eligible "cluster representatives". In many applications $J = I$. However in some applications $|J| < |I|$, i.e. only objects with certain characteristics can qualify as representatives (e.g. survey papers and books in the automatic classification of technical material in some field). In other cases $|I| < |J|$, (e.g. in the automatic classification of technical material, an alternative to the above policy is to represent a cluster by a list of key words. This list does not need to match exactly that of any single paper in the cluster).

The clustering problem is defined relative to a matrix of parameters c_{ij} that represent the similarity between objects i and j. For example c_{ij} could be the number of common key words associated with technical papers i and j. Anderberg (1973) gives several ways of calculating the similarity matrix C. In most applications there are no natural fixed charges. Rather, the constraint (1.6) is added to the formulation (1.1)-(1.4).

2. Brief Historical Overview

There is a vast literature on the uncapacitated location problem. Krarup and Pruzan (1983) give an up-to-date survey. We will not repeat their effort here. However, to set the stage for the ensuing sections, we will mention the main solution approaches and cite some basic references.

The first approaches to solve UL were heuristic. One of the earliest heuristics is due to Kuehn and Hamburger (1963), who actually present it for a wider class of location problems. It consists of two routines. The first routine opens facilities sequentially in an order that maximizes the increase of the objective function at each step. It stops when adding a new facility could only decrease the objective. Kuehn and Hamburger called it the "add routine"; in the modern literature such a procedure is called a greedy heuristic because of its appetite for maximum improvement at each step. Their second routine is the "bump and shift routine". It eliminates (bumps) any facility that has become uneconomical because of the presence of other facilities chosen subsequently by the greedy heuristic. Then, starting from this feasible solution, it considers interchanging (shifting) an open and closed facility. Such a pairwise interchange is performed if it improves the current feasible solution and the procedure stops when a solution has been found that cannot be improved by such interchanges. In the remainder of this chapter, the shifting procedure will be referred to as an interchange heuristic.

The greedy and interchange heuristics are the basis of numerous approximation algorithms. They can, of course, be helpful in exact algorithms that require feasible initial solutions or use feasible solutions in other ways, Spielberg (1969b) and Hansen and Kaufman (1972). However,

when a heuristic is used to obtain a final solution, it is very important to have an upper bound as well so that the user can be confident that the heuristic solution value is not too far from the optimal value. Cornuejols, Fisher and Nemhauser (1977b) gave such bounds for the greedy and interchange heuristics, both a priori worst-case bounds and a posteriori bounds on a particular solution constructed by the heuristic. These bounds were generalized to the maximization of submodular set functions (see Section 9) by Nemhauser, Wolsey and Fisher (1978).

General solution techniques for finding optimal solutions to integer programs have been customized for UL. The mixed integer linear programming formulation can be solved by Benders decomposition, Benders (1962). This approach was proposed by Balinski and Wolfe (1963) and appears to have been the first attempt to solve UL to optimality. The computational experiments were discouraging, see Balinski (1965), and this method was abandoned until recently. Magnanti and Wong (1977) develop techniques to accelerate the convergence of Benders decompositon. They generate strong cuts from the set of feasible Benders cuts and by so doing they are able to reduce the number of integer programs to be solved. Nemhauser and Wolsey (1981) consider the Benders cuts in the more general framework of maximizing a submodular set function.

Branch-and-bound algorithms for the uncapacitated plant location problem use the fact that it is not necessary to constrain the variables y_{ij} to be integral. Branching is done on a binary enumeration tree with respect to the variables x_j. Bounds are obtained from one of the two linear programming relaxations - (1.1), (1.2), (1.5) and (1.3) or (1.3'). The first algorithms

used the linear program with (1.3'). Efroymson and Roy (1966) showed that this linear program can be solved analytically so that the bound at each node of the enumeration tree could be computed very quickly in constant time. Improvements to Efroymson's and Roy's algorithm were made by Spielberg (1969a), Khumawala (1972) and Hansen (1972). However, when (1.3') is used in a linear programming relaxation, the bounds obtained are generally not sufficiently strong to curtail the enumeration adequately.

As we observed previously, the constraints (1.3) imply (1.3') but not conversely. The linear programming relaxation that uses (1.3) is called the strong linear programming relaxation abbreviated by SLP. Revelle and Swain (1970), among others, observed that SLP is so effective that its solution is very often integral. Thus a branch-and-bound algorithm that uses SLP to compute bounds is very likely to perform well in the sense that very little (if any) enumeration will be required. However, because of its size, it is not efficient to solve SLP directly by the simplex method.

Much of the recent research on UL has involved the development of special purpose algorithms for solving SLP. Marsten (1972) used parametric linear programming and a special implementation of the simplex method. Garfinkel, Neebe and Rao (1974) used Dantzig-Wolfe decomposition. Schrage (1975) devised a generalized simplex method to treat the variable upper bounds (1.3). Guignard and Spielberg (1977) suggested a version of the simplex method that pivots only to integral vertices of the polytope (1.2), (1.3), (1.6). Cornuejols and Thizy (1982b) used a primal subgradient algorithm.

Dual algorithms or algorithms that solve the dual of the strong linear programming relaxation have the advantage that upper bounds are obtained from any dual feasible solution. Thus a sufficiently good bound may be obtained to fathom a node of the enumeration tree prior to solving the dual to optimality.

Bilde and Krarup (1977) and Erlenkotter (1978) used heuristic methods to obtain a near-optimal solution of the dual. Erlenkotter went a step further by using the complementarity slackness conditions of linear programming to improve this bound. His procedure was so effective that in 45 out of the 48 problems that he tested, optimality was reached at the first node of the branch-and-bound algorithm. His DUALOC code appears to outperform all existing algorithms.

A Lagrangian dual of the formulation (1.1)-(1.4), proposed by Geoffrion (1974), is obtained by weighting the constraints (1.2) by multipliers and placing them in the objective function. It can be solved using subgradient optimization, see Held, Wolfe and Crowder (1974). The Lagrangian approach can also be used for the p-facility location problem. Some computational results are reported in Narula, Ogbu and Samuelsson (1977), in Cornuejols, Fisher, Nemhauser (1977b) and in Mulvey and Crowder (1979). Krarup and Pruzan (1983) mention a different Lagrangian dual obtained when constraints (1.3) (instead of (1.2)) are weighted by multipliers and placed in the objective function.

3. Computational Complexity

An algorithm is said to be a polynomial-time algorithm for problem P, if for all instances of P (possible data sets), the computing time of the algorithm can be bounded by a polynomial function of the data size. If L measures the data size and k is the order of the polynomial, we say that the computing time of the algorithm is $O(L^k)$. Sometimes it is more convenient to express the computing time as a function of basic data parameters, such as the dimension of a matrix or the number of nodes in a graph. Then, but only then, it is assumed that all arithmetic operations and comparisons are performed in unit time.

A fundamental theoretical question, also of some practical importance, is whether a given combinatorial optimization problem can be solved by some polynomial-time algorithm. Denote by P the class of problems that can be solved in polynomial-time, i.e. by some polynomial-time algorithm. For most combinatorial optimization problems of practical interest, the question - are they in P? - has not been answered. A significant step was made by Cook (1971) and Karp (1972) who introduced the notion of NP-complete problems. This is a class of combinatorial problems that are equivalent in the sense that either all or none of these problems can be solved by a polynomial-time algorithm.

At present no polynomial-time algorithm is known for solving any NP-complete problem and it has been widely conjectured that none exists. A problem is said to be NP-hard if the existence of a polynomial-time algorithm to solve it would imply that all NP-complete problems can be solved by a polynomial-time algorithm. Thus to show that a problem (P) is NP-hard it suffices to find a polynomial transformation that reduces a known NP-complete

problem, see e.g. the comprehensive list given by Garey and Johnson (1979), to the problem (P).

Theorem 3.1 The uncapacitated plant location problem is NP-hard.

Proof: We need to introduce the vertex cover problem:

Given a graph G and an integer k, find whether there exists a subset of k vertices of G that cover all the edges of G. (Vertex v is said to cover edge e if v is an endpoint of e.) The vertex cover problem is NP-complete, see Karp (1972) or Garey and Johnson (1979). We reduce it to UL.

Consider a graph $G = (V,E)$ with vertex set V and edge set E. Construct an instance of UL with the set of potential facilities $J = V$ and set of clients $I = E$. Let $c_{ij} = 0$ for all $i \in E$ and $j \in V$, and let $f_j = 1$ for all $j \in V$. This transformation is polynomial in the size of the graph.

Note that the instance of UL defined in this way consists of covering all the edges of the graph G with the minimum number of vertices. Thus an optimal solution of UL provides the answer to the vertex cover problem. This proves that UL is NP-hard. □

A polynomial transformation that reduces a known NP-hard problem to a problem (Q) shows that (Q) is also NP-hard. An immediate corollary of Theorem 3.1 is that the p-facility location problem is NP-hard since solving it for every $p = 1,\ldots,n$ provides a solution to UL.

Although UL is NP-hard, some special cases can be solved in polynomial-time. Kolen (1982) has shown that UL is solvable in time $O(r^3)$ when the

problem is defined as a tree with r nodes and certain other assumptions are satisfied. These are that the clients as well as the facilities are located at nodes of the tree. A length is associated with each edge of the tree and d_{ij} is the distance between nodes i and j. Kolen solves the formulation (1.1)', (1.2)-(1.4) and shows that the strong linear programming relaxation always has an integral optimal solution.

Another interesting case of the uncapacitated plant location problem that can be solved in polynomial time was discovered by Krarup and Bilde (1977). In this instance too, the crux is that the strong linear programming relaxation always has an integral optimal solution. The conditions required by Krarup and Bilde generalize those obtained when a classical economic lot size problem is formulated as an uncapacitated plant location problem.

Finally, Barany, Edmonds and Wolsey (1983) have given a polynomial-time algorithm for a tree partitioning problem that contains both Kolen's and Krarup's and Bilde's problems.

4. Duality

Suppose we are given a feasible solution to UL that is claimed to be optimal or nearly optimal (within a specified absolute or relative tolerance). We know of only two ways to verify this claim.

a. enumeration: compare, perhaps implicitly, the value of this feasible solution to all others.

b. bounding: determine an upper bound on the optimal value of all feasible solutions that is sharp enough to verify the claim.

Enumeration is useful algorithmically only when it can be done implicitly. Generally, this means that the enumerative approach, as in a branch-and-bound algorithm, uses upper bounds to curtail the enumeration. Conversely, an algorithm whose primary thrust is bounding may need to resort to some enumeration to verify the claim.

The point is that good upper bounds, as well as good feasible solutions, are crucial in solving UL, as for that matter, any hard combinatorial optimization problem. We will see, however, that UL has many features that make it a relatively easy NP-hard problem.

Duality plays a key role in the determination of upper bounds. The dual of the strong linear programming relaxation given by (1.1)-(1.3), (1.5) is

$$W = \min \sum_{i \in I} u_i + \sum_{j \in J} t_j \tag{4.1}$$

$$u_i + w_{ij} \geq c_{ij} \quad \text{all } i \in I, j \in J \tag{4.2}$$

$$-\sum_{i \in I} w_{ij} + t_j \geq -f_j \quad \text{all } j \in J \tag{4.3}$$

$$w_{ij}, t_j \geq 0 \quad \text{all } i \in I, j \in J. \tag{4.4}$$

We can eliminate variables and constraints from this formulation by noting that:

a. for given w_{ij}, (4.1) implies that we would like to make t_j as small as possible. Thus (4.3) and (4.4) imply that $t_j = (\sum_{i \in I} w_{ij} - f_j)^+$, where $(\alpha)^+ = \max(0,\alpha)$.

b. for given u_i, (4.1) also implies that we would like to make w_{ij} as small as possible. Thus (4.2) and (4.4) imply that $w_{ij} = (c_{ij} - u_i)^+$.

Thus

$$t_j = [\sum_{i \in I} (c_{ij} - u_i)^+ - f_j]^+ \quad \text{all } j \in J. \tag{4.5}$$

If we think of the u_i's as prices associated with the clients, the t_j's are the profits from the facilities relative to these prices. In other words, if someone agreed to pay us u_i for the right to serve the ith client, we would be willing to sell the jth facility for the price t_j given by (4.5).

Substituting (4.5) into (4.1) yields the condensed dual

$$W = \min_u \{ \sum_{i \in I} u_i + \sum_{j \in J} [\sum_{i \in I} (c_{ij} - u_i)^+ - f_j]^+ \}. \tag{4.6}$$

The dual is then to determine a minimum sum set of prices u_i for the clients and consequently a minimum sum set of prices for the facilities so that we would agree to sell the operation. It tells us the linear programming approximation of the worth of our assets.

If $\sum_{i \in I} [(c_{ij} - u_i)^+ - f_j]^+ > 0$, then some u_i can be increased without increasing the objective function (4.6). Also if $u_i > \max_{j \in J} c_{ij}$, then u_i

can be decreased without increasing the objective function (4.6). These observations yield a second condensed dual

$$W = \min_{u} \sum_{i \in I} u_i \tag{4.7}$$

$$\sum_{i \in I} (c_{ij} - u_i)^+ - f_j \leq 0 \quad \text{all} \quad j \in J \tag{4.8}$$

$$u_i \leq \max_{j \in J} c_{ij} \qquad \text{all} \quad i \in I. \tag{4.9}$$

Although both condensed dual formulations are nonlinear, they are important because they contain only m variables; furthermore for any value of u, we obtain the upper bound

$$W(u) = \sum_{i \in I} u_i + \sum_{j \in J} [\sum_{i \in I} (c_{ij} - u_i)^+ - f_j]^+. \tag{4.10}$$

When (4.8) holds, the upper bound reduces to

$$W(u) = \sum_{i \in I} u_i. \tag{4.11}$$

A Lagrangian dual of (1.1)-(1.4) is obtained by weighting the constraints (1.2) by multipliers u_i and placing them in the objective function. Let

$$L(u) = \max[\sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j + \sum_{i \in I} u_i (1 - \sum_{j \in J} y_{ij})] \tag{4.12}$$

subject to (1.3) and (1.4).

Frequently, a Lagrangian dual provides a tighter upper bound than a linear programming dual. But this is not the case here.

Proposition 4.2 $L(u) = W(u)$ for all u.

Proof: $L(u) = \max \sum_{i\in I} \sum_{j\in J} (c_{ij} - u_i)y_{ij} - \sum_{j\in J} f_j x_j + \sum_{i\in I} u_i$ subject to (1.3) and (1.4). Hence $y_{ij} = x_j$ if $c_{ij} - u_i > 0$, $y_{ij} = 0$ if $c_{ij} - u_i < 0$ and $y_{ij} = 0$ or x_j if $c_{ij} - u_i = 0$. Thus

$$L(u) = \max_{x_j\in\{0,1\}} \sum_{j\in J} \Big[\sum_{i\in I} (c_{ij} - u_i)^+ - f_j\Big] x_j + \sum_{i\in I} u_i \tag{4.13}$$

$$= \sum_{j\in J} t_j + \sum_{i\in I} u_i = W(u)$$

where t_j is given by (4.5). This is true since $x_j = 1$ if $\sum_{i\in I} (c_{ij} - u_i)^+ - f_j > 0$, $x_j = 0$ if $\sum_{i\in I} (c_{ij} - u_i)^+ - f_j < 0$ and $x_j = 0$ or 1 otherwise. □

Corollary 4.3 $L = \min_u L(u) = W$.

When u satisfies the constraints (4.8), the solution of (4.13) satisfies the complementarity conditions

$$\Big(\sum_{i\in I} (c_{ij} - u_i)^+ - f_j\Big)x_j = 0 \quad \text{all } j \in J. \tag{4.14}$$

Equation (4.14) suggests that if u satisfies (4.8), to find a good primal solution we should only consider opening those facilities for which $\sum_{i\in I} (c_{ij} - u_i)^+ - f_j = 0$.

5. Heuristics

The combinatorial formulation of the uncapacitated plant location problem $\max_{S \subseteq J} z(S)$ where $z(S) = \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j$ can be viewed as a condensed, nonlinear primal that depends only on the values of the sets $S \subseteq J$. Based on this observation numerous authors have proposed heuristics that iterate on the set S of open locations and avoid an explicit integer programming formulation. Two of the most basic heuristic approaches are described below: the greedy and interchange heuristics.

The Greedy Heuristic.

Start with no facilities open. Given a set S of open facilities, open that facility $j \notin S$ whose incremental value $\rho_j(S) = z(S \cup \{j\}) - z(S)$ is as large as possible, and is positive. If no such facility exists stop with the set S of open facilities.

Formally

Initialization $S^0 = \emptyset$, $\rho_j(\phi) = \sum_{i \in I} c_{ij} - f_j$, $t = 1$.

Iteration t Find $j_t = \arg\max_{j \notin S^{t-1}} \rho_j(S^{t-1})$.

If $\rho_{j_t}(S^{t-1}) \leq 0$ and $t > 1$, stop. The set S^{t-1} is the greedy solution with value $Z^G = z(S^{t-1})$. If $t = 1$, the greedy solution is $S^1 = \{j_1\}$.

If $\rho_{j_t}(S^{t-1}) > 0$, $S^t = S^{t-1} \cup \{j_t\}$.

Set $t \leftarrow t + 1$.

The greedy heuristic requires at most n iterations and each iteration requires $O(nm)$ calculations. Thus the overall running time is $O(n^2m)$.

An Example

We will use the following small example to introduce and motivate the ideas developed subsequently. Real-world problems are typically much larger (e.g. $m = n = 100$). Consider the uncapacitated facility location problem defined by the data:

$$m = 4,\ n = 6,\ f = (3,2,2,2,3,3) \text{ and}$$

$$C = \begin{pmatrix} 6 & 6 & 8 & 6 & 0 & 6 \\ 6 & 8 & 6 & 0 & 6 & 6 \\ 5 & 0 & 3 & 6 & 3 & 0 \\ 2 & 3 & 0 & 2 & 4 & 4 \end{pmatrix} .$$

Applying the greedy heuristic to the example yields

iteration 1: $(\rho_1(\phi),\ldots,\rho_6(\phi)) = (16,15,15,12,10,13)$.

Hence $j_1 = 1$ and $S^1 = \{1\}$.

iteration 2: $(\rho_2(\{1\}),\ldots,\rho_6(\{1\})) = (1,0,-1,-1,-1)$.

Hence $j_2 = 2$ and $S^2 = \{1,2\}$.

iteration 3: $(\rho_3(\{1,2\}),\ldots,\rho_6(\{1,2\})) = (0,-1,-1,-1)$.

The set S^2 of value $Z^G = z(S^2) = 17$ is the greedy solution.

We can now use (4.10) to obtain upper bounds on the values of the feasible solutions produced by the greedy heuristic. In fact, such an upper bound can be associated with any $S \subseteq J$.

Define

$$\bar{u}_i(S) = \max_{j \in S} c_{ij} \quad \text{all} \quad i \in S.$$

Note that

$$z(S) = \sum_{i \in I} \bar{u}_i(S) - \sum_{j \in S} f_j$$

and

$$\rho_j(S) = \sum_{i \in I} (c_{ij} - \bar{u}_i(S))^+ - f_j \quad \text{all} \quad j \notin S.$$

Thus, from (4.10)

$$W(\bar{u}(S)) = \sum_{i \in I} \bar{u}_i(S) + \sum_{j \notin S} [\rho_j(S)]^+$$

since $\quad c_{ij} - \bar{u}_i(S) \leq 0 \quad$ for all $\quad i \in I \quad$ and $\quad j \in S.$

In particular if S^G is the final set chosen by the greedy heuristic then, by the stopping criterion, $S_j(S^G) \leq 0 \quad j \notin S^G$ so that $W(\bar{u}(S^G)) = \sum_{i \in I} \bar{u}_i(S^G)$. We have shown that the greedy solution S^G deviates from optimality by at most $\sum_{j \in S^G} f_j$, which suggests that it will yield a small error when the fixed costs are small in comparison to the profits. Furthermore, we may obtain a better bound by considering all of the sets produced by the greedy algorithm. Let $\bar{u}_i^0 = \min_{j \in J} c_{ij}$ all $i \in I$ and $\bar{u}^k = \bar{u}(S^k)$, $k = 1,\ldots,t-1$. Define a dual greedy value by $W^G = \min_k W(\bar{u}^k)$. In the example, $\bar{u}^0 = (0\ 0\ 0\ 0)$, $W(\bar{u}^0) = \sum_{j \in J} \rho_j(\phi) = 83$, $\bar{u}^1 = (6,6,5,2)$,

$W(\bar{u}^1) = \sum_{i \in I} \bar{u}_i^1 + 1 = 20$, $\bar{u}^2 = (6,8,5,3)$ and $W(\bar{u}^2) = \sum_{i \in I} \bar{u}_i^2 = 22$. Hence $W^G = 20$.

The bound we have given so far is an a posteriori bound for a particular instance of UL. In fact, a general relationship between Z^G and W^G that a priori applies to all instances of UL is given by

Theorem 5.1 [Cornuejols, Fisher and Nemhauser (1977b)]

$$Z^G \geq (\frac{e-1}{e})W^G + (\frac{1}{e})R$$

where e is the base of the natural logarithm and

$$R = \sum_{i \in I} \min_{j \in J} c_{ij} - \sum_{j \in J} f_j.$$

A proof of Theorem 5.1 that uses linear programming duality is given in Fisher, Nemhauser and Wolsey [1978].

For the p-facility location problem with $c_{ij} \geq 0$ for all i and j and $f_j = 0$ all $j \in J$, we have $R \geq 0$. Thus we achieve a simple data independent statement of Theorem 5.1.

Corollary 5.2 [Cornuejols, Fisher and Nemhauser (1977)] For the p-facility location problem with $c_{ij} \geq 0$ for all i and j and $f_j = 0$ all $j \in J$

$$\frac{Z^G}{W^G} \geq \frac{e-1}{e} \simeq 0.63.$$

There are families of p-facility location problems for which this bound is achieved asymptotically. Furthermore, since $Z^G \leq Z \leq W \leq W^G$

$$\max\left\{ \frac{Z^G}{Z} , \frac{Z}{W} , \frac{W}{W^G} \right\} \geq \left(\frac{e-1}{e}\right)^{1/3} \cong 0.86.$$

There are several variations and generalizations of the greedy heuristic for which bounds similar to those of Theorem 5.1 and Corollary 5.2 are known, see Cornuejols, Fisher and Nemhauser [1977b] and Nemhauser, Wolsey and Fisher [1978]. For example, we can begin with all facilities open and at each iteration close a facility that gives the largest improvement in the objective function so long as such a facility exists. A generalization of the greedy heuristic is to start with the family consisting of all sets of cardinality k, for some fixed k, and apply greedy to each of these $\binom{n}{k}$ initial sets separately; we then choose the best of the resulting $\binom{n}{k}$ solutions.

None of these variations or generalizations, however, improve the worst-case bound of the greedy heuristic. In fact, what is remarkable about the bound on the greedy heuristic is not its value, but that no polynomial-time procedure of any degree whatsoever is known for p-facility location problems that has a better worst-case performance.

The salient feature of the greedy heuristic is that the maximum possible improvement is made at each step. If this is not done, worst-case performance deteriorates, even if a broader choice of improvements is considered.

An example of such a heuristic is <u>generalized interchange</u>, see Nemhauser, Wolsey and Fisher (1978). Here we begin with an arbitrary set S^0. Given

S^{t-1}, at iteration t we select any set S^t such that $z(S^t) > z(S^{t-1})$, $|S^t \setminus S^{t-1}| \leq 1$ and $|S^{t-1} \setminus S^t| \leq 1$ or stop if no such S^t exists. Thus, at each iteration, we are allowed to open a facility, close a facility or do both so long as an improvement is made.

The worst-case bound of generalized interchange is weaker than that of greedy. For example, under the conditions of Corollary 5.2, this heuristic can guarantee only to find a solution of value at least half of the optimum value. Of course, by starting with a greedy solution, the bounds of Theorem 5.1 and Corollary 5.2 are obtained. Nevertheless, they are not strengthened by applying generalized interchange. On the other hand, greedy followed by generalized interchange seems to give good empirical performance, see Hansen (1972) and Cornuejols, Fisher and Nemhauser (1977b).

The dual solution $\bar{u}(S)$ given by (5.1) is motivated by its use in obtaining a bound on the given primal solution S. Conversely, given a dual solution that satisfies (4.8), the complementarity conditions (4.14) suggest considering a primal solution in which $x_j = 0$ if $\sum_{i \in I} (c_{ij} - u_i)^+ - f_i < 0$. Let

$$J(u) = \{j: \sum_{i \in I} (c_{ij} - u_i)^+ - f_j = 0\}.$$

The best solution that satisfies complementarity is obtained by solving

$$\max_{S \subseteq J(u)} \{ \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j \},$$

but this problem may not be much easier to solve than UL itself. Instead, we take any minimal set $K(u) \subseteq J(u)$ that satisfies

$$\max_{j \in K(u)} c_{ij} = \max_{j \in J(u)} c_{ij} \quad \text{all } i \in I. \tag{5.2}$$

Proposition 5.3 Given a u that satisfies (4.8) and $u_i \leq \max_{j \in J(u)} c_{ij}$ all $i \in I$, and a primal solution $K(u)$ defined by (5.2), let $k_i = |\{j \in K(u): c_{ij} > u_i\}|$. If $k_i \leq 1$ all $i \in I$, then u is an optimal set of open facilities.

Proof:

$$z(K(u)) = \sum_{i \in I} \max_{j \in K(u)} c_{ij} - \sum_{j \in K(u)} f_j.$$

If $k_i = 0$

$$\max_{j \in K(u)} c_{ij} = u_i = u_i + \sum_{j \in K(u)} (c_{ij} - u_i)^+$$

and if $k_i = 1$

$$\max_{j \in K(u)} c_{ij} = u_i + \sum_{j \in K(u)} (c_{ij} - u_i)^+.$$

Hence, if $k_i \leq 1$ all $i \in I$,

$$z(K(u)) = \sum_{i \in I} \sum_{j \in K(u)} (c_{ij} - u_i)^+ - \sum_{j \in K(u)} f_j + \sum_{i \in I} u_i$$

$$= \sum_{j \in K(u)} \left(\sum_{i \in I} (c_{ij} - u_i)^+ - f_j \right) + \sum_{i \in I} u_i$$

$$= \sum_{i \in I} u_i = W(u) \quad \text{by (4.11).} \quad \square$$

In our example, with $u = (6\ 6\ 4\ 3)$, we obtain $J(u) = K(u) = \{2,3,4\}$ and $(k_1,k_2,k_3,k_4) = (1\ 1\ 1\ 0)$. Hence these are optimal primal and dual solutions of value $\sum_{i \in I} u_i = 19$.

While Proposition 5.3 may permit us to recognize an optimal solution, it is limited to those cases in which $\min_u W(u) = Z$ and even then it is still necessary to find an appropriate u and $K(u)$.

Dual Descent

Dual descent is a heuristic that begins with a u satisfying (4.8) and then attempts to decrease the u_i's one-at-a-time while maintaining (4.8), see Erlenkotter (1978) and Bilde and Krarup (1977). It is surprisingly effective, but not fail safe, in finding a u that satisfies the conditions of Proposition 5.3. This descent approach, with some embellishments, is the inner loop of Erlenkotter's DUALOC algorithm. The basic descent method procedes as follows:

Begin with $u_i^0 = \max_{j \in J} c_{ij}$ all $i \in I$. Cycle through the indices $i \in I$ one-by-one attempting to decrease u_i to the next smaller value of c_{ij}. If one of the constraints

$$\sum_{i \in I} (c_{ij} - u_i)^+ \leq f_j \quad \text{all} \quad j \in J \tag{5.3}$$

blocks the decrease of u_i to the next smaller c_{ij}, u_i is decreased to the minimum value allowed by the constraint. When all of the u_i's are blocked from further decreases, the procedure terminates.

The reason for decreasing u_i only to the next smaller c_{ij}, rather than to the smallest permissible value, is to keep the k_i of Proposition 5.3 as small as possible.

Applying dual descent to our example yields the results shown in Table 5.1. For the first four steps, each

Step	u^T				$f_j - \sum_{i \in I} (c_{ij} - u_i)^+$					
0	8	8	6	4	3	2	2	2	3	3
1	6	8	6	4	3	2	0	2	3	3
2	6	6	6	4	3	0	0	2	3	3
3	6	6	5	4	3	0	0	1	3	3
4	6	6	5	3	3	0	0	1	2	2
5	6	6	4	3	2	0	0	0	2	2

Table 5.1

of the u_i's is decreased to the second max in the row. Now u_1 is considered again, but cannot be decreased because the constraints (5.3) for $j = 3$ would be violated. Similarly, a decrease of u_2 would violate (5.3) for $j = 2$. However, u_3 can be decreased; but it is decreased only to 4 because (5.3) becomes active for $j = 4$ when $u_3 = 4$. Finally u_4 cannot be decreased because of (5.3) for $j = 2$. This completes the dual descent with $u = (6\ 6\ 4\ 3)$ and $W(u) = \sum_{i \in I} u_i = 19$. Now, as noted above, $J(u) = K(u) = \{2,3,4\}$ and we also obtain a primal solution of value 19.

A possible improvement of dual descent, which is likelier to produce a primal and dual pair for which Proposition 5.3 applies, is obtained by modifying the order in which the u_i's are considered as candidates to decrease. In particular, rather than just cycling through the u_i's, let $Q_i(u) = \{j: c_{ij} - u_i \geq 0\}$. Then if u_i is decreased and descent terminates,

$k_i \leq |Q_i(u)|$. Hence we choose u_s next if $|Q_s(u)| \leq |Q_i(u)|$ for all $i \in I$.

Suppose dual descent terminates with the dual solution u^* and we determine a primal solution given by $K(u^*)$ such that Proposition 5.3 fails to verify optimality. Then there exists an i such that $k_i > 1$. By increasing u_i to its previous value and reapplying dual descent, it may be possible to improve the dual solution further.

We have sketched all of the basic steps of Erlenkotter's heuristic. An example in which it does not find an optimal solution to (4.6) is given by

$$f = (2\ 2\ 2), \qquad C = \begin{pmatrix} 2 & 2 & 0 \\ 2 & 0 & 2 \\ 0 & 2 & 2 \end{pmatrix}. \tag{5.4}$$

In this instance of UL, beginning with $u^0 = (2\ 2\ 2)$, dual descent terminates with $u = (0\ 2\ 2)$ and the embellishments don't help. However an optimal u is $(1\ 1\ 1)$ yielding $W = 3$ and $Z = 2$. Nevertheless, this heuristic has performed extremely well on the problems that Erlenkotter has considered. He reports that in 45 of 48 problems tested, the heuristic found an optimal solution. To provide the capability of finding an optimal solution and proving optimality, the heuristic is imbedded in a branch-and-bound algorithm called DUALOC. Given its simplicity, speed and availability, DUALOC may be the most efficient way to solve UL. However, it could bog down on hard problems in which the heuristic bound is not as good as the linear programming bound. Thus one is motivated to develop efficient algorithms for solving the strong linear programming relaxation.

6. Algorithms and Reformulations of the Strong Linear Programming Relaxation

The strong linear programming relaxation (SLP) of the uncapacitated facility location problem (UL) is

$$(6.1) \qquad Z_{LP} = \max \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j$$

$$(6.2) \qquad \sum_{j \in J} y_{ij} = 1 \qquad \text{all } i \in I$$

$$(6.3) \qquad y_{ij} - x_j \leq 0 \qquad \text{all } i \in I, j \in J$$

$$(6.4) \qquad y_{ij} \geq 0, \; x_j \geq 0 \qquad \text{all } i \in I, j \in J.$$

For general integer programs, a linear programming relaxation must be used in conjunction with cutting planes or enumeration to obtain an optimal integral solution. However, for reasons which are barely understood, SLP is an unusually powerful relaxation of UL in the following sense.

Observation 6.1 Very frequently, SLP has an optimal integral solution.

Of course, it is not true that all of the extreme points of the polyhedron (6.2)-(6.4) are integral. Example (5.4) has a unique optimal solution with $x = (1/2\ 1/2\ 1/2)$ and it is easy to construct infinite families of objective functions for which the unique optimal solution to (6.1)-(6.4) is fractional. Nevertheless, randomly constructed objective functions and the few encountered in practice that have appeared in the literature strongly support Observation 6.1.

A challenging problem for the combinatorial mathematician is to make Observation 6.1 precise. For the practitioner, Observation 6.1 means that

an efficient method for solving SLP will also be an efficient method for solving most instances of UL. Even if one is unlucky, a very good upper bound is typically attained so that a branch-and-bound algorithm should terminate rapidly.

Even relatively modestly sized instances of UL cannot be solved by a standard mixed integer programming package that uses SLP as a linear programming relaxation because of the large number of constraints (6.3). For example, $m = n = 100$ yields a problem with more than 10,000 constraints. In this section, we consider several approaches for solving very large structured linear programs. We will begin by briefly mentioning two direct approaches to eliminating the difficulty caused by the large number of constraints (6.3). Then we will apply some well-known reformulations and algorithms including Lagrangian duality and subgradient optimization, Dantzig-Wolfe decomposition, Benders decomposition and subgradient optimization on the primal. Connections among those approaches will be noted. Finally, we will consider a reformulation that involves the reduction of the matrix C into an interesting canonical form.

Direct Approaches

The constraints $y_{ij} \leq x_j$ are generalizations of simple upper bound constraints in which the upper bounds themselves are variables. It is well-known how to handle fixed upper bound constraints in the simplex method without expanding the dimension of the basis to include them. Schrage (1975) has generalized this idea to incorporate variable upper bounds. He reports computational results obtained by applying his method to SLP.

An alternative is to generate the constraints $y_{ij} \leq x_j$ as cuts only when they are violated in an optimal solution to the weak linear programming relaxation. This idea has been tested by Morris (1978). In the example of the last section, an optimal solution to the weak linear programming relaxation is $x_2 = x_3 = x_4 = x_5 = 1/4$ and $y_{13} = y_{22} = y_{34} = y_{45} = 1$. We could then add the four violated variable upper bound constraints and continue to solve the linear program.

The direct approaches are primal. Dual methods may be superior for two reasons. First, if SLP is incorporated in a branch-and-bound algorithm, it may not be necessary to solve SLP to optimality at every node of the enumeration tree, i.e. at some of the nodes, a dual feasible solution may suffice to bound the subproblem. Second, as we have already shown in the last section, we can easily generate an integral primal solution from each dual solution.

Lagrangian Duality and Subgradient Optimization

The Lagrangian $L(u)$ of (4.12) forms the basis of a subgradient algorithm for solving the dual of SLP. The subgradient algorithm solves the problem $\min_u L(u)$.

The function $L(u)$ given by (4.12) is the maximum of a finite number of linear functions. Therefore $L(u)$ is piecewise linear and convex. Subgradient optimization [Held, Wolfe and Crowder (1974)] has proved to be a useful method for minimizing unconstrained piecewise linear convex functions. This approach is an extension of the gradient method for minimizing smooth, nonlinear convex functions. Since gradients do not exist at non-differentiable

points of $L(u)$, the gradient direction is replaced by a subgradient direction, which will be explained below.

Given u^t, an iteration of the subgradient algorithm generates a new dual solution by the formula

$$u^{t+1} = u^t - \gamma^t \partial L(u^t) \tag{6.5}$$

where $\partial L(u^t)$ is a subgradient at u^t and γ^t is the stepsize. If $\partial L(u^t) = 0$, then u^t is an optimal dual solution.

Suppose

$$L(u) = \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij}^* - \sum_{j \in J} f_j x_j^* + \sum_{i \in I} u_i \left(1 - \sum_{j \in J} y_{ij}^*\right)$$

where $\{x_j^*, y_{ij}^*\}$ are defined in the proof of Proposition 4.2. If the $\{y_{ij}^*\}$ are unique then

$$\partial L(u)_i = 1 - \sum_{j \in J} y_{ij}^* \quad \text{all} \quad i \in I \tag{6.6}$$

is the gradient of $L(u)$ at u. However if the $\{y_{ij}^*\}$ are not unique, then any direction given by (6.6) or convex combinations of such directions is a subgradient direction. Although a step in a subgradient direction does

not guarantee a decrease in $L(u)$, it can be proved that with an appropriate choice of stepsize the iterates given by (6.5) converge to an optimal solution [Polyak (1969)]. Cornuejols, Fisher and Nemhauser (1977b) have solved the Lagrangian dual by subgradient optimization and report computational results.

In our example, if we start with $u^0 = (8\ 8\ 6\ 4)$, then $L(u^0) = 26$ and $\partial L(u) = (1\ 1\ 1\ 1)$. With a stepsize of $\gamma^0 = 2$, we obtain $u^1 = (6\ 6\ 4\ 2)$ and $L(u^1) = 19$. The solution to (4.13) is not unique; however, the solution $x_2 = x_4 = 1$, $x_j = 0$ otherwise and $y_{12} = y_{22} = y_{34} = y_{42} = 1$, $y_{ij} = 0$ otherwise yields $\partial L(u^1) = (0\ 0\ 0\ 0)$ and verifies the optimality of u^1.

Dantzig-Wolfe Decomposition

For all non-empty $R \subseteq I$, let $\lambda_j^R = 1$ if facility j serves only these clients in the set R and $\lambda_j^R = 0$ otherwise. If $\lambda_j^R = 1$, facility j yields a profit of $\sum_{i \in R} c_{ij} - f_j$. Thus UL can be reformulated as the integer program

$$(6.7) \qquad Z = \max \sum_{j \in J} \sum_{R \subseteq I} \left(\sum_{i \in R} c_{ij} - f_j \right) \lambda_j^R$$

$$(6.8) \qquad \sum_{j \in J} \sum_{R \ni i} \lambda_j^R = 1 \qquad \text{all } i \in I$$

$$(6.9) \qquad \sum_{R \subseteq I} \lambda_j^R \leq 1 \qquad \text{all } j \in J$$

$$(6.10) \qquad \lambda_j^R \in \{0,1\} \quad \text{all } R \subseteq I,\ j \in J.$$

The equations (6.8) state that each client is served by exactly one facility and the inequalities (6.9) state that each facility can serve only one set of clients.

We are going to study the linear programming relaxation of this integer program where (6.10) is replaced by

$$\lambda_j^R \geq 0 \quad \text{all} \quad R \subseteq I,\ j \in J. \tag{6.11}$$

Proposition 6.2 Let $\bar{Z}$ be the optimal value of the linear program (6.7), (6.8), (6.9) and (6.11). Then $\bar{Z} = Z_{LP}$.

Proof Apply Dantzig-Wolfe decomposition to the linear program (6.1)-(6.4) with master constraints (6.2) and subproblem constraints (6.3), (6.4) and $x_j \leq 1$ all $j \in J$. Substituting the subproblem extreme points into (6.2) yields (6.8), and (6.9) are the convexity constraints for the subproblems. □

Before considering an algorithm, we will make some simplifications. If $\sum_{i \in R} (c_{ij} - f_j) > \sum_{i \in R} (c_{ik} - f_k)$, then $\lambda_k^R = 0$ in every optimal solution. Hence for each R, we need only one variable, say λ_R with price

$$d_R = \sum_{i \in R} c_{ij(R)} - f_{j(R)} = \max_{j \in J} \left(\sum_{i \in R} c_{ij} - f_j \right).$$

Furthermore the constraints (6.9) are superfluous. This is true because if $R \cap R' \neq \phi$, then (6.8) guarantees that $\lambda_R + \lambda_{R'} \leq 1$ and if $R \cap R' = \phi$ and

$j(R) = j(R') = k$, then $f_k \geq 0$ implies $d_{R \cup R'} \geq d_R + d_{R'}$.

Thus we can restate the integer program as

$$Z = \max \sum_{R \subseteq I} d_R \lambda_R \tag{6.12}$$

$$\sum_{R \ni i} \lambda_R = 1 \qquad \text{all } i \in I \tag{6.13}$$

$$\lambda_R \in \{0,1\} \quad \text{all } R \subseteq I. \tag{6.14}$$

This formulation has 2^{m-1} variables and m constraints. Curiously, n does not enter into the size of the problem so that for fixed m, this formulation can be solved in polynomial-time.

The linear program (6.12), (6.13) and

$$\lambda_R \geq 0 \quad \text{all } R \subseteq I \tag{6.15}$$

can be solved by column generation. Suppose we begin with m columns, say those for $R = \{i\}$, $i = 1,\ldots,m$. Then the primal solution is $\lambda_R^0 = 1$ for $R = \{i\}$, $i = 1,\ldots,m$, and the dual solution is $u_i^0 = \max_{j \in J}(c_{ij} - f_j)$ all $i \in I$.

We now see if any of the nonbasic columns have a positive price. This can be done at iteration p by solving for each $j \in J$ the subproblem

$$t_j^p = \max \sum_{i \in I} (c_{ij} - u_i^p) y_{ij} - f_j x_j \tag{6.16}$$

(6.17) $$y_{ij} - x_j \leq 0 \qquad \text{all } i \in I$$

(6.18) $$y_{ij} \in \{0,1\},\ x_j \in \{0,1\} \quad \text{all } i \in I$$

since $\sum_{i \in R} (c_{ij} - u_i^p) - f_j$ is the price of variable λ_j^R. We obtain $t_j^p = (\sum_{i \in I} (c_{ij} - u_i^p)^+ - f_j)^+$. If $t_j^p = 0$ all $j \in J$ then the current solution is optimal. For each k such that $t_k^p > 0$, let R_k be any set satisfying $\{i: c_{ik} - u_i^p > 0\} \subseteq R_k \subseteq \{i: c_{ik} - u_i^p \geq 0\}$. We now add to the linear program the variables λ_{R_k} for all k such that $t_k^p > 0$.

Garfinkel, Neebe and Rao (1974) have obtained computational experience with this type of algorithm.

An important feature of this approach is that both lower and upper bounds on Z_{LP} are obtained at each iteration. By primal feasibility, $Z_{LP} \geq \sum_{i \in I} u_i^p$; and from (4.12) we see that $Z_{LP} \leq \sum_{j \in J} t_j^p + \sum_{i \in I} u_i^p = W(u^p)$. Moreover, as well as obviously being a primal method, it is also a dual method that can be compared with solving the Lagrangian dual by subgradient optimization. Here the u_i's at each iteration are determined by solving a linear program, while in the previous method the dual variables are determined by moving in the direction of a subgradient of the function $L(u)$. An advantage of the column generation approach is that its lower bounds are determined in a less ad hoc fashion. Furthermore, whenever the linear program has an integral solution a feasible solution to the integer program is also found.

In our example, we start with an initial basis consisting of the four unit columns $R_i = \{i\}$, $i = 1,\ldots,4$ with objective coefficients $d_{R_1} = d_{R_2} = 6$,

$d_{R_3} = 4$ and $d_{R_4} = 1$. This yields the dual solution $u^0 = (6\ 6\ 4\ 1)$. Solving the subproblems, we obtain $t^0 = (0\ 2\ 0\ 1\ 0\ 0)$ and thus $17 \leq Z_{LP} \leq 20$. We generate two new columns $R_5 = \{1\ 2\ 4\}$ and $R_6 = \{3\ 4\}$ with objective coefficients $d_5 = 15$ and $d_6 = 6$. The next master linear program yields the primal solution $\lambda^1_{R_5} = \lambda^1_{R_3} = 1$, $\lambda^1_{R_i} = 0$ otherwise and the dual solution $u^1 = (6\ 6\ 4\ 3)$. Now $t^1_j = 0$ all $j \in J$ so that the primal has been solved. In terms of the original variables, we have $x_2 = x_4 = 1$, $x_j = 0$ otherwise.

Primal Subgradient Algorithm

With fixed x_j's, $0 \leq x_j \leq 1$ all $j \in J$, $\sum_{j \in J} x_j \geq 1$, SLP is

$$Z_{LP}(x) = -\sum_{j \in J} f_j x_j + \max \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \tag{6.19}$$

$$\sum_{j \in J} y_{ij} = 1 \quad \text{all } i \in I \tag{6.20}$$

$$y_{ij} \leq x_j \quad \text{all } i \in I \ \ j \in J \tag{6.21}$$

$$y_{ij} \geq 0 \quad \text{all } i \in I, \ j \in J. \tag{6.22}$$

Let $V_i(x)$ be the profit from the ith client. Then (6.19)-(6.22) decomposes as follows

$$Z_{LP}(x) = -\sum_{j \in J} f_j x_j + \sum_{i \in I} V_i(x) \tag{6.23}$$

where for each $i \in I$

$$V_i(x) = \max \sum_{j \in J} c_{ij} y_{ij} \tag{6.24}$$

$$\sum_{j \in J} y_{ij} = 1 \tag{6.25}$$

$$y_{ij} \leq x_j \quad \text{all } j \in J \tag{6.26}$$

$$y_{ij} \geq 0 \quad \text{all } j \in J. \tag{6.27}$$

This linear program can be solved greedily. Let $c_{ij_1} \geq c_{ij_2} \geq \cdots \geq c_{ij_n}$. Then $y_{ij_k} = x_{j_k}$ $k = 1,\ldots,p-1$, $y_{ij_p} = 1 - \sum_{k=1}^{p-1} x_{j_k}$, $y_{ij_k} = 0$ otherwise where $\sum_{k=1}^{p-1} x_{j_k} < 1 \leq \sum_{k=1}^{p} x_{j_k}$.

The dual of (6.24)-(6.27) is for each $i \in I$

$$V_i(x) = \min u_i + \sum_{j \in J} x_j w_{ij} \tag{6.28}$$

$$u_i + w_{ij} \geq c_{ij} \quad \text{all } j \in J \tag{6.29}$$

$$w_{ij} \geq 0 \quad \text{all } j \in J. \tag{6.30}$$

To solve (6.28)-(6.30), observe that $w_{ij} = (c_{ij} - u_i)^+$ and that it sufficies to consider $u_i \in \{c_{i1},\ldots,c_{in}\}$. Hence

$$V_i(x) = \min_{k \in J} (c_{ik} + \sum_{j \in J} x_j (c_{ij} - c_{ik})^+) \quad \text{all } i \in I. \tag{6.31}$$

Since $V_i(x)$ is the minimum of a finite number of linear functions, it is a piecewise linear and concave function. Therefore $Z_{LP}(x)$ is also piecewise linear and concave and

$$Z_{LP} = \max Z_{LP}(x)$$

$$0 \leq x_j \leq 1 \quad \text{all} \quad j \in J \tag{6.32}$$

$$\sum_{j \in J} x_j \geq 1$$

can be solved by subgradient optimization. If there is a j such that $\sum_{i \in I} c_{ij} - f_j > 0$, the constraint $\sum_{j \in J} x_j \geq 1$ is superfluous. If not, we can add a constant $M > \min_{j \in J}(f_j - \sum_{i \in I} c_{ij})$ to some row of matrix C without changing the solution while assuring that $\sum_{j \in J} x_j \geq 1$ will be satisfied by any optimal solution. In the remainder of the chapter, we assume that $\sum_{j \in J} x_j \geq 1$ is not needed.

A subgradient of $Z_{LP}(x)$ at x is of the form

$$\partial V(x)_j = \sum_{i \in I} (c_{ij} - c_{ip})^+ - f_j \quad \text{all} \quad j \in J \tag{6.33}$$

where p is determined from a solution to the ith subproblem (6.24)-(6.27), i.e. $c_{ip} = \min_j \{c_{ij} : y_{ij} > 0\}$. Note that because of the bounds on the variables, if $x_j = 0$ then $\partial V(x)_j$ is replaced by $[\sum_{i \in I} (c_{ij} - c_{ip})^+ - f_j]^+$, and if $x_j = 1$ then $\partial V(x)_j$ is replaced by $\min(0, \sum_{i \in I} (c_{ij} - c_{ip})^+ - f_j)$.

Cornuejols and Thizy (1982b) report their computational experience in solving (6.32) by a subgradient algorithm.

The results of attempting to solve (6.32) by subgradient optimization for our example are summarized in Table 6.1. An optimal solution to SLP is obtained at iteration 3, but the subgradient formula (6.33) is not adequate to verify it.

iteration	x						V(x)				$Z_{LP}(x)$	∂V(x)						step size
0	0	1	1	1	1	0	8	8	6	4	17	-3	-2	-2	-2	-3	-3	1/4
1	0	1/2	1/2	1/2	1/4	0	7	7	9/2	3	18	-1	1	0	1	-1	-1	1/4
2	0	3/4	1/2	3/4	0	0	7	15/2	21/4	11/4	37/2	-1	1	0	-1	-1	-1	1/4
3	0	1	1/2	1	0	0	7	8	6	3	19	-3	-2	0	-2	-2	-2	

Table 6.1

Benders Decomposition

An alternative way of using (6.23) and (6.31) is to formulate the linear program

$$(6.34) \qquad Z_{LP} = \max \sum_{i \in I} V_i - \sum_{j \in J} f_j x_j$$

$$(6.35) \qquad V_i - \sum_{j \in J} (c_{ij} - c_{ik})^+ x_j \leq c_{ik} \quad \text{all } i \in I,\ k \in J$$

$$(6.36) \qquad 0 \leq x_j \leq 1 \quad \text{all } j \in J.$$

This is precisely the linear program that arises when applying Benders decomposition to SLP. But now we have mn constraints of the form (6.35). However, we can think of these as cutting planes and generate them only as we need them.

In particular, suppose we have only a proper subset of the constraints (6.35). We solve the relaxed linear program and determine an optimal solution (x^q,V^q). Now we use x^q in (6.24)-(6.27) to determine $V_i(x^q)$ all $i \in I$. If

$$V_i(x^q) \leq V_i^q \quad \text{all} \quad i \in I, \tag{6.37}$$

then (x^q,V^q) satisfies all of the constraints (6.35) and x^q is an optimal solution. If not, each i for which (6.37) is violated specifies a violated constraint of the form (6.35). These are added to the linear program and we continue.

In our example, we begin with the constraints (6.35) determined by the second maximum in each row, i.e., $V_1 - 2x_3 \leq 6$, $V_2 - 2x_2 \leq 6$, $V_3 - x_4 \leq 5$, and $V_4 \leq 4$. A solution to the linear program is $V^1 = (6\ 8\ 5\ 4)$ and $x^1 = (0\ 1\ 0\ 0\ 0\ 0)$. Then by solving (6.24)-(6.27), we obtain $V(x^1) = (6\ 8\ 0\ 3)$ and generate the constraints

$$V_3 - 5x_1 - 3x_3 - 6x_4 - 3x_5 \leq 0$$

$$V_4 \qquad\qquad - x_5 - x_6 \leq 3.$$

Now we obtain the solution $V^2 = (6\ 8\ 6\ 3)$ and $x^2 = (0\ 1\ 0\ 1\ 0\ 0)$. Since $V(x^2) = (6\ 8\ 6\ 3)$, all of the constraints of (6.35) are satisfied and x^2 is an optimal solution.

Magnanti and Wong (1977) have used a variation of this approach and have developed stronger inequalities in an attempt to impose integrality on x.

Canonical Reduction

We now develop another formulation that involves the disaggregation and aggregation of clients, see Cornuejols, Nemhauser and Wolsey (1980). The aggregation of two clients i_1 and i_2 means to replace clients i_1 and i_2 by a single client i such that

$$c_{ij} = c_{i_1 j} + c_{i_2 j} \quad \text{all } j \in J. \tag{6.38}$$

The disaggregation of client i into two clients i_1 and i_2 means to replace i by two clients i_1 and i_2 such that (6.38) holds. While aggregation is uniquely defined, disaggregation is not.

In general, aggregation yields an underestimation of profit and disaggregation overestimates. This is true, because if (6.38) is satisfied the greedy solution to the linear program (6.24)-(6.27) implies $V_i(x) \leq V_{i_1}(x) + V_{i_2}(x)$ for all x such that $0 \leq x_j \leq 1$. We say that aggregation or disaggregation is valid when for all x such that $0 \leq x_j \leq 1$

$$V_i(x) = V_{i_1}(x) + V_{i_2}(x). \tag{6.39}$$

Proposition 6.3 Let s_k be a permutation of $\{1,2,\ldots,n\}$ such that $c_{ks_k(1)} \geq c_{ks_k(2)} \geq \cdots \geq c_{ks_k(n)}$.

a. If i_1 and i_2 have permutations such that $s_{i_1(j)} = s_{i_2(j)}$, $j = 1,\ldots,n$ then (6.38) is a valid aggregation.

b. If i, i_1 and i_2 have permutations such that $s_{i(j)} = s_{i_1(j)} = s_{i_2(j)}$, $j = 1,\ldots,n$ then (6.38) is a valid disaggregation.

Proof a. If $s_{i_1(j)} = s_{i_2(j)}$ for all j and rows i_1 and i_2 are aggregated by (6.38), then $s_{i(j)} = s_{i_1(j)} = s_{i_2(j)}$ $j = 1,\ldots,n$. Thus the greedy solution to (6.24)-(6.27) implies that (6.39) holds.

b. The proof is similar to that of a. and is left as an exercise. □

The following proposition shows how a row can be disaggregated.

Proposition 6.4 Suppose $c_{is_i(1)} \geq \cdots \geq c_{is_i(p-1)} > c_{is_i(p)} \geq \cdots \geq c_{is_i(n)}$. Then for $j = 1,\ldots,p-1$

$$c_{i_1 s_{i_1}}(j) = c_{i s_i}(p) \quad \text{and} \quad c_{i_2 s_{i_2}}(j) = c_{i s_i}(j) - c_{i s_i}(p)$$

and for $j = p,\ldots,n$

$$c_{i_1 s_{i_1}}(j) = c_{i s_i}(j) \quad \text{and} \quad c_{i_2 s_{i_2}}(j) = 0$$

is a valid disaggregation of row i.

Proof The condition of Proposition 6.3b holds. □

We can apply Proposition 6.4 recursively to disaggregate a row i into at most n rows, $i_1,\ldots,i_n$ with the following properties: (i) $c_{i_t j} \in \{0, r_{i_t}\}$ for $t = 1,\ldots,n$, (ii) $r_{i_t} > 0$ for $t > 1$, (iii) $c_{i_1 j} = r_{i_1}$ for $j = 1,\ldots,n$, (iv) $c_{i_t j} = 0$ implies $c_{i_{t+1} j} = 0$ for $t = 2,\ldots,n-1$.

To do this, suppose $c_{i s_i}(q) = \ldots = c_{i s_i}(n)$ for some $q \leq n$.

Apply Proposition 6.4 with $p = q$. This yields

$$c_{i_1 s_{i_1}}(j) = c_{i s_i}(q), \qquad j = 1,\ldots,n$$

$$c_{i_2 s_{i_2}}(j) = \begin{cases} c_{i s_i}(j) - c_{i s_i}(q) & j = 1,\ldots,q-1 \\ 0 & j = q,\ldots,n. \end{cases}$$

Row i_1 is in the desired form and if $c_{i s_i}(j) = c_{i s_i}(j+1)$, $j = 1,\ldots,q-2$ so is row i_2. Otherwise let ℓ be the largest value of j for which

$c_{is_i(\ell-1)} > c_{is_i(\ell)}$. Now apply Proposition 6.4 with $p = \ell$ to disaggregate row i_2. Here we refer to the two new rows as i_2 and i_3. Thus

$$c_{i_2 s_{i_2}(j)} = \begin{cases} c_{is_i(\ell)} - c_{is_i(q)} & j = 1,\ldots,q-1 \\ 0 & j = q,\ldots,n \end{cases}$$

$$c_{i_3 s_{i_3}(j)} = \begin{cases} c_{is_i(j)} - c_{is_i(\ell)} & j = 1,\ldots,\ell-1 \\ 0 & j = \ell,\ldots,n. \end{cases}$$

Row i_2 is now in the desired form and we now disaggregate row i_3 if necessary. Since at each step, the row to be disaggregated has at least one more zero than the previous row, the procedure takes at most $n - 1$ steps and yields at most n rows each having the desired property.

Consider

$$(c_{i1}\ c_{i2}\ c_{i3}\ c_{i4}) = (4\ 2\ 2\ -1).$$

We obtain

$$(c_{i_1 1}\ c_{i_1 2}\ c_{i_1 3}\ c_{i_1 4}) = (-1\ -1\ -1\ -1)$$

and

$$(c_{i_2 1}\ c_{i_2 2}\ c_{i_2 3}\ c_{i_2 4}) = (5\ 3\ 3\ 0).$$

Disaggregating row i_2 yields (3 3 3 0) and (2 0 0 0). Thus a client represented by the row (4 2 2 -1) can be replaced by 3 equivalents clients: (-1 -1 -1 -1), (3 3 3 0) and (2 0 0 0).

Suppose each of the clients $i \in I$ is disaggregated in this way. Now we may obtain pairs of clients say i_t and k_ℓ such that $c_{i_t j} \neq 0$ and $c_{p_\ell j} \neq 0$ if and only if $j \in T$. By Proposition 6.3. a., these two clients can be aggregated into a single client with profit $c_{i_t j} + c_{p_\ell j}$ for $j \in T$ and 0 profit for $j \notin T$. Finally, a client whose profit is constant for all $j \in J$ can be eliminated from the problem since this client produces the same profit for all feasible x.

To summarize this discussion, we can transform the matrix C into an equivalent <u>canonical</u> matrix R containing at most $\min(m(n-1), 2^n-2)$ rows. Each row of R represents a set $T = J \setminus \phi$ and there is a profit r_T for all $j \in T$ and a profit of zero for $j \notin T$.

In our example,

$$R = \begin{pmatrix} 6 & 6 & 6 & 6 & 0 & 6 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 6 & 6 & 6 & 0 & 6 & 6 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 3 & 3 & 3 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \text{client 1} \\ \\ \text{client 2} \\ \\ \text{client 3} \\ \\ \\ \text{client 4} \\ \\ \end{matrix}$$

We will now use this transformation to obtain a reformulation of SLP. Given x, $0 \leq x_j \leq 1$ all $j \in J$, the problem for the client that represents the set T that is equivalent to (6.24)-(6.27) is

$$V_T(x) = r_T(\min(\sum_{j \in T} x_j, 1)) = r_T - r_T(1 - \sum_{j \in T} x_j)^+.$$

Let π_T be the fraction of client T not served. Then

$$V_T(x) = r_T + \max(-r_T\pi_T)$$

$$\pi_T \geq 1 - \sum_{j \in T} x_j$$

$$\pi_T \geq 0.$$

Let be the collection of subsets of J that are represented in the profit matrix R. Then

$$Z_{LP} = \sum_{T \in \mathcal{T}} r_T + \max(-\sum_{T \in \mathcal{T}} r_T\pi_T - \sum_{j \in J} f_j x_j) \tag{6.40}$$

$$\pi_T + \sum_{j \in T} x_j \geq 1 \quad \text{all } T \in \mathcal{T} \tag{6.41}$$

$$\pi_T \geq 0,\ x_j \geq 0 \quad \text{all } T \in \mathcal{T},\ \text{all } j \in J. \tag{6.42}$$

We rewrite the objective function as

$$Z^*_{LP} = Z_{LP} - \sum_{T \in \mathcal{T}} r_T = - \min\left(\sum_{T \in \mathcal{T}} r_T \pi_T + \sum_{j \in J} f_j x_j \right) . \tag{6.43}$$

The dual of (6.41)-(6.43) is

$$Z^*_{LP} = - \max \sum_{T \in \mathcal{T}} u_T \tag{6.44}$$

$$\sum_{T \ni j} u_T \leq f_j \qquad \text{all } j \in J \tag{6.45}$$

$$0 \leq u_T \leq r_T \qquad \text{all } T \in \mathcal{T} . \tag{6.46}$$

The linear program (6.44)-(6.46) has at most $m(n-1)$ variables and n constraints plus upper bounds on the variables. It is the most compact linear programming formulation we know and has the structure of the linear programming relaxation of a set packing problem. In our example, see the matrix R given above, there are only 10 variables and 6 constraints other than upper bounds. In comparison, the original linear programming formulation of SLP ((6.1)-(6.4)) has 32 variables and 28 constraints. In experimenting with some k-median problems, Cornuejols, Nemhauser and Wolsey (1980) have observed that when the simplex method (with upper bounds treated implicitly) is applied to the various linear programming formulations, the formulation (6.44)-(6.46) was by far the best one in terms of simplex pivots and running time. In addition, the formulation (6.44)-(6.46) provides a nice interpretation for the dual descent heuristic given in Section 5. We leave this as an exercise.

Summary

This section has emphasized the description of exact algorithms for the solution of SLP. These algorithms, and effective heuristics for the dual, such as dual descent, can be viewed as subroutines that must be embedded within a branch-and-bound code based on enumeration of the 0-1 variables x_j.

A second viewpoint is based on a specific MIP (mixed integer programming) formulation that can, in principle, be tackled by any general purpose mixed integer programming package. Here candidates are the original formulation (1.1)-(1.4), the Benders formulation (6.34)-(6.36) with $x_j \in (0,1)$ all $j \in J$, and the new canonical formulation (6.40)-(6.42) with $x_j \in \{0,1\}$ all $j \in J$. Such formulations are also amenable to treatment by algorithms other than branch-and-bound--see for instance the cutting planes for (1.1)-(1.4) developed in the next section. However, attempting to fit any of these formulations into a general purpose package leads to difficulties of problem size since each of these formulations involves $O(mn)$ constraints and/or variables.

Having been inundated with various heuristics and algorithms, the reader has the right to ask for a recommendation on the appropriate method to use. Unfortunately there is no simple answer.

Among the special purpose branch-and-bound algorithms DUALOC appears to be the best. It is a generally available easy to use FORTRAN program that is very fast on most problems.

If one insists on solving the strong linear program to optimality, which may be necessary for the harder problems, solving the Lagrangian dual by subgradient optimization provides an easy to program and relatively fast approach. If the simplex algorithm is to be used, the linear program (6.44)-(6.46) has a significant advantage in size, and limited computational experience suggests that it is best in terms of time and pivots.

The main advantage of using a general purpose MIP code is that complicating constraints create no difficulties, whereas a special purpose code becomes unusable. In addition, work is in progress on general purpose codes which will be capable of working with the compact formulation $\sum y_{ij} \leq mx_j$ and generating violated variable upper bound constraints $y_{ij} \leq x_j$ as needed; see Martin and Schrage (1982) and Van Roy and Wolsey (1983). This should permit such codes to handle medium sized UL's as general MIP's.

7. Polyhedral Results

Extreme Points of SLP

We have observed that the strong linear programming relaxation

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} y_{ij} - \sum_{j=1}^{n} f_j x_j$$

subject to

$$(7.1) \qquad \begin{cases} \sum_{j=1}^{n} y_{ij} = 1 & i = 1,\ldots,m \\ 0 \le y_{ij} \le x_j \le 1 & i = 1,\ldots,m \text{ and } j = 1,\ldots,n, \end{cases}$$

very often has integral optimal solutions. Although this phenomenon is not well understood, some properties of the polytope (7.1) are known. When $m \le 2$ or $n \le 2$, it has been shown by Muckendi (1975), Krarup and Pruzan (1983) and Cho et al. (1983) that all the extreme points of (7.1) are integral. In fact, the constraint matrix is totally unimodular in that case. However, even when $m = n = 3$ the strong linear programming relaxation may have fractional optimal solutions. For example, when $C = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ and $f_j = 1$ for $j = 1,2,3$, we have remarked previously that $x_j = 1/2$ for $j = 1,2,3$ and $y_{ij} = 1/2$ for $i \neq j$, 0 for $i = j$, where $i, j = 1,2,3$ is the unique optimal solution.

The fractional extreme points of (7.1) are completely characterized by the next theorem. For a given non-integral solution (x,y) of (7.1) let

$J_1 = \{j \in J: 0 < x_j < 1\}$ and

$I_1 = \{i \in I: y_{ij} = 0 \text{ or } x_j \text{ for all } j \text{ and } y_{ij} \text{ is fractional for some } j\}$.

Let $a_{ij} = 1$ if $y_{ij} > 0$ and 0 otherwise, and denote by A the $|I_1| \times |J_1|$ matrix whose elements are a_{ij} for $i \in I_1$, $j \in J_1$.

Theorem 7.1 [(Cornuejols, Fisher and Nemhauser (1977a)]. A fractional solution (x,y) of (7.1) is an extreme point of (7.1) if and only if

(i) $x_j = \max_{i \in I} y_{ij}$ for all $j \in J_1$

(ii) for each $i \in I$, there is at most one $j \in J$ with $0 < y_{ij} < x_j$

(iii) the rank of A equals $|J_1|$.

The 3 conditions of this theorem are easily verified for the example given above.

Since the polyhedron defined by (7.1) has many fractional extreme points, the type of objective function that is optimized over this polyhedron must play an important role in the attainment of integral optimal solutions. Frequently, $C = (c_{ij})$ is defined over a network with the property that $c_{ij} = -d_i(t_{ij} + q_j)$ decreases as a function of i, the further i is from j in the network. (I.e. if i' is on the shortest path from i to j in the network then $c_{ij} \leq c_{i'j}$). The influence of this property on the solution of SLP for tree networks will be discussed in Section 8. For more general graphs, it is an interesting open question.

When $C = (c_{ij})$ is a general 0,1 matrix and $f_j = 1$ all $j \in J$, UL is the problem of finding the minimum number of columns that cover all the rows of the maxtrix C. (A set S of columns covers row i if $c_{ij} = 1$ for at least one $j \in S$.) This problem is known as the set covering problem and often has fractional optimal solutions.

Some valid inequalities for UL that remove fractional extreme points of the SLP polytope are given in the next theorem.

Theorem 7.2 Cho et al. (1983). Let B be a k x k nonsingular 0,1 matrix such that $B^{-1} e \geq 0$, where e is a column vector of ones. Index the rows and columns of B by $I_k \subseteq I$ and $J_k \subseteq J$ where $|I_k| = |J_k| = k$. Then

$$\sum_{i \in I_k} \sum_{j \in J_k} b_{ij} y_{ij} - \sum_{j \in J_k} x_j \leq \lfloor k - e^T B^{-1} e \rfloor$$

is a valid inequality for UL. It cuts off at least one fractional extreme point of the polytope (7.1) if $e^T B^{-1} e$ is not integral.

For example, if $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ we generate the constraint $y_{12} + y_{13} + y_{21} + y_{23} + y_{31} + y_{32} - x_1 - x_2 - x_3 \leq 1$, which cuts off the fractional extreme point given above.

Conversely, it is easy to show that the family of valid inequalities defined in Theorem 7.2 cuts off all the fractional extreme points of the polytope (7.1). However, in general new fractional extreme points arise.

We now turn to the identification of valid inequalities for UL that define facets of the integer polytope.

Facets of the Integer Polytope

Let $P_{m,n}$ be the polytope defined as the convex hull of the integer solutions to the system (7.1)

$$\begin{cases} \sum_{j=1}^{n} y_{ij} = 1 & i = 1,\ldots,m \\ 0 \leq y_{ij} \leq x_j \leq 1 & i = 1,\ldots,m \text{ and } j = 1,\ldots,n \\ x_j,\ y_{ij} \in \{0,1\} & i = 1,\ldots,m \text{ and } j = 1,\ldots,n. \end{cases}$$

A set of $k + 1$ points $w^0, w^1,\ldots,w^k$ are affinely independent if the k vectors $w^1 - w^0,\ldots,w^k - w^0$ are linearly independent. A polytope has dimension k if it contains $k + 1$ affinely independent points but not more. An affine space is the intersection of hyperplanes. The smallest affine space which contains a polytope P is called its affine hull. The polytope $P_{m,n}$ has dimension $mn + n - m$ and affine hull

$$\{(x,y) \in R^n \times R^{mn} : \sum_{j=1}^{n} y_{ij} = 1 \text{ for } i = 1,\ldots,m\}.$$

A face of a polytope P is a set $F = P \cap \{x: ax = b\}$ where $ax \leq b$ is satisfied by every $x \in P$ and $ax < b$ for at least one $x \in P$. The inequality $ax \leq b$ is said to define the face F. Any face of a polytope is itself a polytope. When its dimension is one less than that of the polytope P, the face F is called a facet. To describe the polytope P by a linear system, it suffices to have a description of its affine hull and one defining inequality for each facet of P.

If such a description of $P_{m,n}$ were known then, in principle, UL could be solved as a linear program since the extreme points of $P_{m,n}$ are precisely the feasible solutions of UL. However, a complete linear system defining $P_{m,n}$ is not known explicitly. Even if one were, only relevant portions of it would be generated to solve an instance of UL, i.e., the facets of $P_{m,n}$ would be used as cutting planes in the spirit of Padberg and Hong's (1980) and Grotschel's (1980) work on the traveling salesman problem.

Whatever the algorithmic use of a partial linear description of $P_{m,n}$ the first step is to identify some of its facets.

Theorem 7.3 The following inequalities define distinct facets of $P_{m,n}$

(i) $y_{ij} \leq x_j$ for all $i \in I$, $j \in J$

(ii) $y_{ij} \geq 0$ for all $i \in I$, $j \in J$

(iii) $x_j \leq 1$ for all $j \in J$.

These facets are called the elementary facets of $P_{m,n}$. The next theorem provides a necessary and sufficient condition for an inequality with coefficients of 0 or 1 to define a facet of $P_{m,n}$.

Assume that $I' \subseteq I$ and $J' \subseteq J$ are two nonempty sets and that $B = (b_{ij})$ $i \in I'$, $j \in J'$ is a 0,1 matrix with no zero row. Consider the inequality

$$(7.2) \qquad \sum_{i \in I'} \sum_{j \in J'} b_{ij} y_{ij} - \sum_{j \in J'} x_j \leq r.$$

Define the graph G as follows. It has a node associated with each variable y_{ij}, $i \in I$, $j \in J$, and x_j, $j \in J$. We will use the same notation for a node and its associated variable. For all $i \in I$ and $j \in J$, the node y_{ij} is joined by an edge to the node x_j and to every node y_{ik} for $k \neq j$.

Let N' be the set of nodes $\{y_{ij}\}$ $i \in I'$, $j \in J'$ $\cup$ $\{x_j\}$ $j \in J'$ and let G' be the subgraph of G induced by the node set N'. Given a graph H we denote by $\alpha(H)$ the maximum size of a stable set in H (a <u>stable set</u> is a set of mutually nonadjacent nodes). Finally, an edge e of H is <u>critical</u> if $\alpha(H - e) > \alpha(H)$, where $H - e$ denotes the graph obtained from H by removing the edge e.

<u>Theorem 7.4</u> [Cornuejols and Thizy (1982a)] The inequality (7.2) is a facet of $P_{m,n}$ if and only if the following set of conditions is satisfied

(i) $r = \alpha(G') - |J|$

(ii) G' is connected,

(iii) for every $i \in I'$, $j \in J'$ such that $b_{ij} = 1$, the edge (x_j, y_{ij}) is critical,

(iv) for every $j, k \in J'$, there exists a sequence of critical edges

$$(y_{i_1 j}, y_{i_1 \ell_1}), (y_{i_2 \ell_1}, y_{i_2 \ell_2}), \ldots, (y_{i_{s-1} \ell_{s-2}}, y_{i_{s-1} \ell_{s-1}}), (y_{i_s \ell_{s-1}}, y_{i_s k}).$$

(v) for every $i \in I$, $j \in J$ such that $y_{ij} \notin N'$, the inequality $\alpha(G') < \alpha(G'')$ is strict, where G'' denotes the subgraph of G induced by $N' \cup \{y_{ij}\}$.

These necessary and sufficient conditions can be used to prove the next theorem, which provides constructively a large class of facets for the uncapacitated plant location polytope.

For $2 \leq t < \ell \leq n$, define $B^{\ell t}_{ij} = (b^{\ell t}_{ij})$ as a matrix with $\binom{\ell}{t}$ rows and ℓ columns whose rows consist of all distinct 0,1 vectors with t ones and $\ell - t$ zeros.

Theorem 7.5 [Cornuejols and Thizy (1982a)] For any pair of integers ℓ and t such that $2 \leq t < \ell \leq n$ and $\binom{\ell}{t} \leq m$, and any sets $I' \subseteq I$, $J' \subseteq J$ such that $|I'| = \binom{\ell}{t}$ and $|J'| = \ell$, the inequality

$$\sum_{i \in I'} \sum_{j \in J'} b^{\ell t}_{ij} y_{ij} - \sum_{j \in J'} x_j \leq \binom{\ell}{t} + t - \ell - 1$$

defines a facet of $P_{m,n}$.

For example, take $t = 2$, $\ell = 3$, $B^{23} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, and $I' = J' = \{1,2,3\}$. According to Theorem 7.5 we get the facet

$$y_{12} + y_{13} + y_{21} + y_{23} + y_{31} + y_{32} - x_1 - x_2 - x_3 \leq 1,$$

which is identical to the valid inequality of Theorem 7.2 that we obtained above with $B = B^{23}$.

Other facets obtained by lifting odd holes or circulant matrices can be found in [Cornuejols and Thizy (1982a)]. Additional material can be found in Cho, Padberg and Rao (1983).

8. Polynomially Solvable Cases

By a special case of UL, we mean a problem of the form (1.1)-(1.4) all of whose instances are described by a subfamily of objective functions (C,f). In this section, we consider two special cases of UL that have the following significant properties.

1. SLP always has an integral optimal solution
2. The problem can be solved in polynomial-time.

A. Economic Lot Sizing

There is a demand d_i in period i, $i = 1,\ldots,n$. The fixed cost of producing in period j is $f_j \geq 0$. The variable production cost is p_j. The variable storage and backorder costs are $c_j^+ \geq 0$ and $c_j^- \geq 0$ respectively. In UL, y_{ij} now represents the fraction of the demand of period i produced in period j, and $x_j = 1$ if and only if there is production in period j,

$$c_{ij} = -(p_j + c_j^+ + \ldots + c_{i-1}^+)d_i \quad \text{if} \quad i \geq j$$

and

$$c_{ij} = -(p_j + c_j^- + \ldots + c_{i+1}^-)d_i \quad \text{if} \quad i < j.$$

B. The Tree Location Problem

Let $G = (V,E)$ be a graph with node set V and edge set E and suppose that G is a tree, i.e. there is a unique path in G between each pair of nodes (G is connected and acyclic). Here the nodes represent both clients and facilities. The cost of opening the jth facility is $f_j \geq 0$ all $j \in V$. Associated with each edge $e \in E$, there is a given non-negative distance. The

distance between nodes i and j is d_{ij} = sum of the edge distances along the unique path between i and j, for all $i, j \in V$. There is also a non-negative scaling function w_i associated with each node i. Let $c_{ij} = -w_i d_{ij}$, $i, j \in V$, $i \neq j$ and $c_{ii} = 0$ all $i \in V$.

The induced subgraph of G generated by $V_j \subseteq V$ is the graph $G_j = (V_j, E_j)$ where $E_j = \{e \in E$: both end nodes of e are in $V_j\}$. G_j is said to be a subtree of G if G_j itself is a tree.

Theorem 8.1 [Kolen (1982)] There is an optimal solution to the tree location problem in which the set of open facilities $S \subseteq V$ is such that for each $j \in S$, $V_j = \{i \in V$: node i is served by node $j\}$ induces a subtree. Moreover, this solution is also optimal to the linear programming relaxation of the tree network problem.

A similar result applies to the lot sizing problem. Consider the tree $G = (V,E)$ where $V = \{1,2,\ldots,n\}$ and $E = \{(i,i+1)$: $i = 1,\ldots,n-1\}$. Here G is simply a path from node 1 to node n so that $V' \subset V$ is a subtree or a path if and only if $V' = \{i, i+1,\ldots,k\}$ for some i and k, $1 \leq i \leq n$ and $k \geq i$.

Theorem 8.2 [Krarup and Bilde (1977)] There is an optimal solution to the lot sizing problem in which the set of periods having positive production $S \subseteq V$ is such that for each $j \in S$, $V_j = \{i \in V$: period i is served by production in period $j\}$ induces a path. Moreover, this solution is also optimal to the linear programming relaxation of the tree network problem.

The fact that an optimal solution to these problems induces subtrees that partition V is not surprising. In the tree location problem, suppose that

node i is on the path joining nodes j and k and node j serves node k but not node i. Suppose node i is served by node $j' \neq j$. Then the cost from this part of the solution is $d_{jk} + f_j + d_{j'1} + f_{j'}$. If instead, node i is served by node j, the cost is $d_{ji} + d_{jk} + f_j + f_{j'}$.

Thus if $d_{ji} \leq d_{j'i}$, we can serve i and all nodes after i that are being served by j' without increasing the cost. This reduces the number of subtree violations by one. Otherwise $d_{ji} > d_{j'i}$, which implies

$$d_{jk} = d_{ji} + d_{ik} > d_{j'i} + d_{ik} = d_{j'k}.$$

This inequality implies that the solution in which j' serves k and only nodes after k that are currently being served by j costs less than the original. It also reduces the number of subtree violations by at least one.

A similar argument proves the result for the economic lot sizing problem.

Both of these results suggest that the ability to partition the solution into subtrees is crucial and leads us to consider the following generalization.

C. <u>The Tree Partitioning Problem</u> Given a tree graph $G = (V,E)$ and a node by node matrix with elements γ_{ij} all $i, j \in V$, let the weight of a subtree $G_j = (V_j,E_j)$ be $w(G_j) = \max_{k \in V_j} \left(\sum_{i \in V_j} \gamma_{ik} \right)$. Find a partition of G into subtrees such that the sum of the weights over all subtrees in the solution is maximum.

To model the lot sizing problem as a tree partitioning problem we take

$$\gamma_{jj} = -(f_j + p_j d_j)$$

$$\gamma_{ij} = -(p_j + c_j^+ + \ldots + c_{i-1}^+)d_i \quad \text{if} \quad i > j$$

$$\gamma_{ij} = -(p_j + c_j^- + \ldots + c_{i+1}^-)d_i \quad \text{if} \quad i < j.$$

To model the tree location problem as a tree partitioning problem we take $\gamma_{jj} = -f_j$ and $\gamma_{ij} = -w_i d_{ij}$ if $i \neq j$.

We now formulate the tree partitioning problem as an integer program in a manner that establishes its connection to UL. If $j^* \in V_j$ is such that $\arg\max_{k \in V_j} (\sum_{i \in V_j} \gamma_{ik}) = j^*$, we say that j^* is the root of subtree G_j. Let $y_{ij} = 1$ if $i \in V$ is in a subtree rooted at j and $y_{ij} = 0$ otherwise. Then the tree partitioning problem can be formulated as

$$\max \sum_i \sum_j \gamma_{ij} y_{ij} \tag{8.1}$$

$$\sum_j y_{ij} = 1 \qquad \text{all } i \in V \tag{8.2}$$

$$y_{ij} - y_{i'j} \leq 0 \qquad \text{all } i, i', j \in V \text{ such that } i' \text{ precedes } i \text{ on a path from } j \text{ to } i \tag{8.3}$$

$$y_{ij} \in \{0,1\} \qquad \text{all } i, j \in V. \tag{8.4}$$

Constraint (8.3) guarantees that if j is the root of a tree that contains i then the tree must also contain i'. Constraint (8.2) guarantees that each node must be in exactly one tree.

The linear programming relaxation of this integer program is obtained by replacing (8.4) by

$$y_{ij} \geq 0 \quad \text{all } i, j \in V. \tag{8.5}$$

Theorem 8.3 [Barany, Edmonds and Wolsey (1983)] The polyhedron defined by (8.2), (8.3) (8.5) has only integral extreme points. Hence for any objective function (8.1) the solution to the linear programming relaxation is integral.

This model for the tree partitioning problem resembles the model (1.1)-(1.4) for UL if we think of the y_{jj}'s as x_j's and replace (8.3) by (1.3). In fact, we can represent solutions of UL as a collection of subtrees that partition a graph G, but unfortunately G is not a tree. In the graph of Figure 8.1, V_1 represents the set of clients and V_2 the set of facilities.

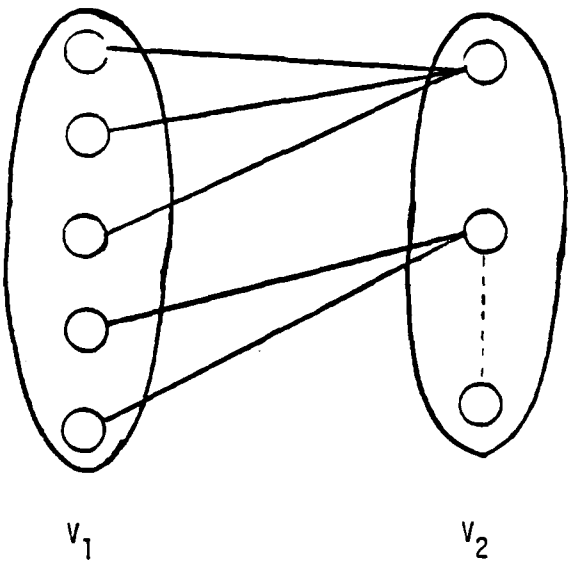


Figure 8.1

A solution to the problem is a set of subtrees, each of which is rooted at a node in V_2. The edges from the root of a tree to nodes in V_1 (solid edges) show the clients being served by the facility that corresponds to the root. A node in V_2 that is not a root corresponds to an unused facility (dashed edges). We set $\gamma_{ij} = c_{ij}$ for $i \in V_1$ and $j \in V_2$. To eliminate the possibility of nodes in V_1 being roots we set $\gamma_{jj} = -M$ for $j \in V_1$ (M is a large positive number) and to represent the fixed costs we set $\gamma_{jj} = -f_j$ for $j \in V_2$. Finally to accommodate unused facilities we set $\gamma_{jk} = 0$ if j and k are in V_2.

We close this section by giving an $O(|V^2|)$ dynamic programming algorithm for solving the tree partitioning problem. Another very general dynamic programming algorithm for location problems on trees can be found in Megiddo, Zemel and Hakimi (1983).

Given the tree $G = (V,E)$ we choose an arbitrary root r. This induces a partial order on V. For all $v \in V$, let $V(v) = \{t: v$ is on the unique path between r and $t\}$ and $S(v) = \{t: v$ is the node that precedes t on the unique path between r and $t\}$. Let T_v be the tree induced by $V(v)$ and $g(v) =$ optimal weight partition for the tree T_v.

The idea of the algorithm is to calculate $g(r)$ recursively by determing $g(v)$ from the $g(w)$ for all $w \in S(v)$. A node w is said to be a leaf of T_r if $S(w) = \emptyset$. Note that we can begin the recursion with $g(w) = \gamma_{ww}$ for all leaves. Before giving the general recursion equation, we need one more definition. Let $g_u(v) =$ optimal weight partition of T_v when v is served by node u and u may not necessarily be an element of $V(v)$. Then

$$g(v) = \max_{u \in V(v)} g_u(v) \tag{8.6}$$

and if w is a leaf of T_r

$$g_u(w) = \gamma_{wu} \quad \text{for all } u \in V. \tag{8.7}$$

Now suppose we are given $g_u(w)$ all $w \in S(v)$ and all $u \in V$. The calculation of $g_u(v)$ divides into two cases as shown in Figure 8.2

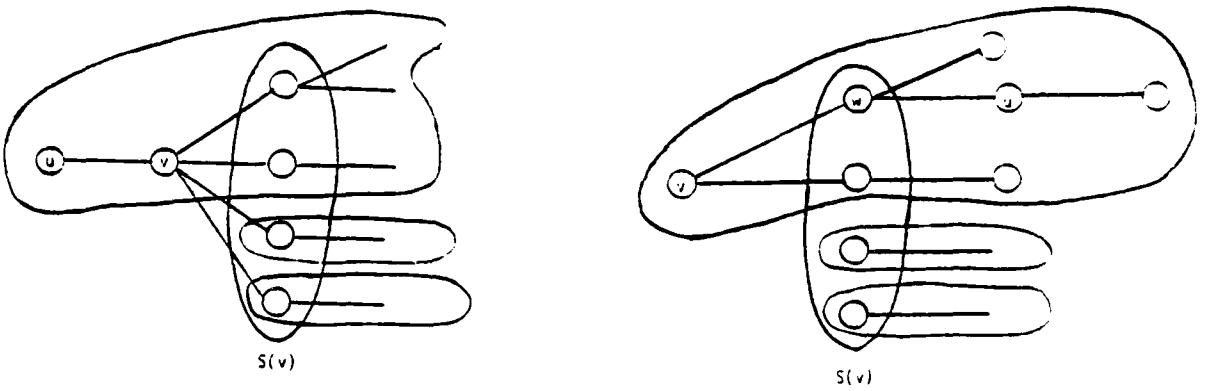


Figure 8.2

If $u \notin V(v)$ or $u = v$ and u serves v, then u will also serve $w \in S(v)$ if $q_u(w) \geq g(w)$. Hence

$$g_u(v) = \gamma_{vu} + \sum_{w \in \tilde{S}(v)} \max\{g_u(w), g(w)\}. \tag{8.8}$$

If $u \in V(v)\backslash\{v\}$, then $u \in V(w^*)$ for some $w^* \in S(v)$. Hence if u serves v then u serves w^*. Thus

$$g_u(v) = \gamma_{vu} + \sum_{w \in \tilde{S}(v)\backslash\{w^*\}} \max\{g_u(w), g(w)\} + g_u(w^*). \tag{8.9}$$

An Example

We consider a tree location problem on the graph of Figure 8.3. The numbers on the edges are the

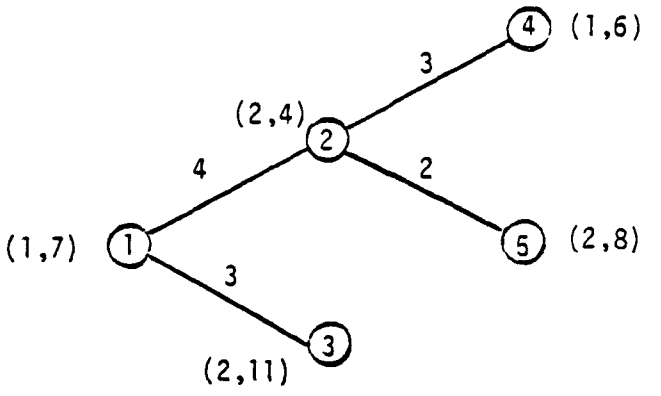


Figure 8.3

d_{ij}'s and the pair of numbers adjacent to the nodes are (w_i, f_i) all $i \in V$. Let $\gamma_{jj} = -f_j$ all $j \in V$ and $\gamma_{ij} = -w_i d_{ij}$ all $i \neq j \in V$. Hence we obtain the matrix

$$\gamma = \begin{pmatrix} -7 & -4 & -3 & -7 & -6 \\ -8 & -4 & -14 & -6 & -4 \\ -6 & -14 & -11 & -20 & -18 \\ -7 & -3 & -10 & -6 & -5 \\ -12 & -4 & -18 & -10 & -8 \end{pmatrix}.$$

For $w = 3,4,5$ the bottom 3 rows of the matrix γ give $g_u(w)$, see (8.7). For node 2, (8.8) and (8.9) yield

$$g_1(2) = \gamma_{21} + \max\{g_1(4), g(4)\} + \max\{g_1(5), g(5)\}$$
$$-8 + \max(-7,-6) + \max(-12,-8) = -22$$

$$g_2(2) = -4 + \max(-3,-6) + \max(-4,-8) = -11$$

$$g_3(2) = -14 + \max(-10,-6) + \max(-18,-8) = -28$$

$$g_4(2) = \gamma_{24} + \max\{g_4(5), g(5)\} + g(4)$$
$$= -6 + \max\{-10,-8\} + (-6) = -20$$

$$g_5(2) = -4 + \max\{-5,-6\} - (-8) = -17.$$

Hence

$$g(2) = \max\{g_2(2), g_4(2), g_5(2)\} = \max\{-11, -20, -17\} = g_2(2) = -11.$$

For node 1 we obtain

$$g_1(1) = -7 + \max\{-22, -11\} + \max\{-6, -11\} = -24$$

$$g_2(1) = -4 + \max\{-14, -11\} + (-11) = -26$$

$$g_3(1) = -3 + \max\{-28, -11\} + (-11) = -25$$

$$g_4(1) = -7 + \max\{-20, -11\} + (-20) = -38$$

$$g_5(1) = -6 + \max\{-18, -11\} + (-17) = -34 .$$

Hence $g(1) = g_1(1) = -24$, where $g_1(1) = \gamma_{11} + g(2) + g_1(3)$.

Thus node 1 serves itself and node 3. Since $g(2) = g_2(2) = \gamma_{22} + g_2(4) + g_2(5)$, node 2 serves itself and nodes 4 and 5.

9. Submodularity

As defined at the outset of this chapter, UL is the combinatorial problem

$$\max_{S \subseteq J} z(S)$$

where

$$z(S) = \sum_{i=1}^{m} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j \tag{9.1}$$

is the profit made when the set S of facilities is open. A very important property of the set function z is its submodularity, A function w defined on the subsets of a finite set J is submodular if

$$w(S \cup \{k\}) - w(S) \leq w(R \cup \{k\}) - w(R) \quad \text{for all} \quad k \notin S \quad \text{and} \quad R \subseteq S \subseteq J - \{k\}.$$

The fact that the profit function z is submodular was observed by Spielberg (1969a), Babayev (1974), Frieze (1974) and Fisher, Nemhauser and Wolsey (1978). It means that the additional profit that can be made by opening a facility in location k when a set S is already open in other locations is a nonincreasing function of S with respect to set inclusion. The larger S, the smaller the profit of establishing a new facility. This is proved formally in the next theorem.

Theorem 9.1 The profit function z is submodular.

Proof Let $R \subseteq S \subseteq J - \{k\}$. For all $i = 1,\ldots,m$

$$\max_{j\in S\cup\{k\}} c_{ij} - \max_{j\in S} c_{ij} = \max(0,\ c_{ik} - \max_{j\in S} c_{ij})$$

$$\leq \max(0,\ c_{ik} - \max_{j\in R} c_{ij}) = \max_{j\in R\cup\{k\}} c_{ij} - \max_{j\in R} c_{ij}$$

where the inequality follows from $\max_{j\in S} c_{ij} \geq \max_{j\in R} c_{ij}$.

By summing these inequalities for all i, we obtain

$$\sum_{i=1}^{m} \max_{j\in S\cup\{k\}} c_{ij} - \sum_{i=1}^{m} \max_{j\in S} c_{ij} \leq \sum_{i=1}^{m} \max_{j\in R\cup\{k\}} c_{ij} - \sum_{i=1}^{m} \max_{j\in R} c_{ij}.$$

Hence

$$z(S\cup\{k\}) - z(S) \leq z(R\cup\{k\}) - z(R). \quad \square$$

Thus UL is a special case of the more general problem

$$\max_{S\subseteq J} \{z(S):\ z \text{ submodular}\}. \tag{9.2}$$

We can apply the greedy and interchange heuristics to (9.2), we can formulate (9.2) as an integer program and many of the results that we have given for UL extend to (9.2) and, in particular, to the capacitated location problem which is another special case of it. We will not elaborate on these results here, but refer the interested reader to Fisher, Nemhauser and Wolsey (1978), Nemhauser and Wolsey (1978), Nemhauser, Wolsey and Fisher (1978), Cornuejols, Nemhauser and Wolsey (1980) and Nemhauser and Wolsey (1981).

References

M.R. Anderberg (1973) Cluster Analysis for Applications, Academic Press, New York.

D.A. Babayev (1974) "Comments on a Note of Frieze," Mathematical Programming 7, 249-252.

E. Balas and M.W. Padberg (1976) "Set Partitioning: A Survey," SIAM Review 18, 710-760.

M.L. Balinski (1965) "Integer Programming: Methods, Uses, Computation," Management Science 12, 253-313.

M.L. Balinski and P. Wolfe (1963) "On Benders Decomposition and a Plant Location Problem," Working Paper ARO-27, Mathematica.

I. Barany, J. Edmonds and L.A. Wolsey (1983), to appear.

M.P. Beck and J.M. Mulvey (1982) "Constructing Optimal Index Funds," Report EES-82-1, School of Engineering and Applied Science, Princeton University.

J.F. Benders (1962) "Partitioning Procedures for Solving Mixed Variables Programming Problems," Numerische Mathematik 4, 238-252.

O. Bilde and J. Krarup (1977) "Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem," Annals of Discrete Mathematics 1, 79-97.

Businessweek (1974) "Making Millions by Stretching the Float," November 23, 89-90.

D.C. Cho, E.L. Johnson, M.W. Padberg and M.R. Rao (1983) "On the Uncapacitated Plant Location Problem I: Valid Inequalities and Facets," Mathematics of Operations Research.

D.C. Cho, M.W. Padberg and M.R. Rao (1983) "On the Uncapacitated Plant Location Problem II: Facets and Lifting Theorems," Mathematics of Operations Research.

S.A. Cook (1971) "The Complexity of Theorem-Proving Procedures," Proceedings 3rd Annual ACM Symposium on the Theory of Computing, 151-158.

G. Cornuejols, M.L. Fisher and G.L. Nemhauser (1977a) "On the Uncapacitated Location Problem," Annals of Discrete Mathematics 1, 163-177.

G. Cornuejols, M.L. Fisher and G.L. Nemhauser (1977b) "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms," Management Science 23, 789-810.

G. Cornuejols, G.L. Nemhauser and L.A. Wolsey (1980) "A Canonical Representation of Simple Plant Location Problems and its Applications," SIAM Journal on Algebraic and Discrete Methods 1, 261-272.

G. Cornuejols and J.M. Thizy (1982a) "Some Facets of the Simple Plant Location Polytope," Mathematical Programming 23, 50-74.

G. Cornuejols and J.M. Thizy (1982b) "A Primal Approach to the Simple Plant Location Problem," SIAM Journal on Algebraic and Discrete Methods 3, 504-510.

M.A. Efroymson and T.L. Ray (1966) "A Branch and Bound Algorithm for Plant Location," Operations Research 14, 361-368.

D. Erlenkotter (1978) "A Dual-based Procedure for Uncapacitated Facility Location," Operations Research 26, 992-1009.

M.L. Fisher, G.L. Nemhauser and L.A. Wolsey (1978) "An Analysis of Approximations for Maximizing Submodular Set Functions II," Mathematical Programming Study 8, 73-87.

A.M. Frieze (1974) "A Cost Function Property for Plant Location Problems," Mathematical Programming 7, 245-248.

M.R. Garey and D.S. Johnson (1979) Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco.

R.S. Garfinkel, A.W. Neebe and M.R. Rao (1974) "An Algorithm for the M-median Plant Location Problem," Transportation Science 8, 217-236.

A.M. Geoffrion (1974) "Lagrangian Relaxation for Integer Programming," Mathematical Programming Study 2, 82-114.

M. Grötschel (1980) "On the Symmetric Travelling Salesman Problem: Solution of a 120-city Problem," Mathematical Programming Study 12, 61-77.

M. Guignard and K. Spielberg (1977) "Algorithms for Exploiting the Structure of the Simple Plant Location Problem," Annals of Discrete Mathematics 1, 247-271.

P. Hansen (1972) "Two Algorithms for the Simple Plant Location Problem Using Additive Penalties," presented at the European Congress of the Econometric Society, Budapest.

P. Hansen and L. Kaufman (1972) "An Algorithm for Central Facilities Location under an Investment Constraint," in P. Van Moeseke ed., Mathematical Programs for Activity Analysis, North Holland, Amsterdam.

M. Held, P. Wolfe and H.P. Crowder (1974) "Validation of Subgradient Optimization," Mathematical Programming 6, 62-88.

R.M. Karp (1972) "Reducibility among Combinatorial Problems," in R.E. Miller and J.W. Thatcher eds., Complexity of Computer Computations, Plenum Press, New York, 85-104.

B.M. Khumawala (1972) "An Efficient Branch and Bound Algorithm for the Warehouse Location Problem," Management Science 18, 718-731.

A. Kolen (1982) "Location Problems on Trees and in the Rectilinear Plane," Mathematisch Centrum, Amsterdam.

J. Krarup and O. Bilde (1977) "Plant Location, Set Covering and Economic Lot Sizing: An O(mn) Algorithm for Structured Problems," in L. Collatz et al eds., Optimierung bei graphentheoretishen and ganzzahligen probleme, Birkhauser Verlag, Basel, 155-180.

J. Krarup and P.M. Pruzan (1983) "The Simple Plant Location Problem: Survey and Synthesis," European Journal of Operations Research 12, 36-81.

A. Kraus, C. Janssen and A. McAdams (1970) "The Lock-box Location Problem, a Class of Fixed Charge Transportation Problems," Journal of Bank Research 1, 51-58.

A.A. Kuehn and M.J. Hamburger (1963) "A Heuristic Program for Locating Warehouses," Management Science 9, 643-666.

T.L. Magnanti and R.T. Wong (1981) "Accelerated Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria," Operations Research 29, 464-484.

R.E. Marsten (1972) "An Algorithm for Finding Almost All the Medians of a Network," Discussion Paper 23, The Center for Mathematical Studies in Econometrics and Management Science, Northwestern University.

K. Martin and L. Shrage (1982) "Some Cuts for 0/1 Mixed Integer Programming," Mimeo, University of Chicago.

N. Megiddo, E. Zemel and S.L. Hakimi (1983) "Maximum Coverage Location Problem," SIAM Journal on Algebraic and Discrete Methods 4, 253-261.

J.G. Morris (1978) "On the Extent to which Certain Fixed-Charge Depot Location Problems can be Solved by LP," Journal of the Operational Research Society 29, 71-76.

C. Mukendi (1975) "Sun l'Implantation d' Equipement dans un Reseau: le Probleme de m-centre," University of Grenoble, France.

J.M. Mulvey and H.L. Crowder (1979) "Cluster Analysis: An Application of Lagrangian Relaxation," Management Science 25, 329-340.

S.C. Narula, U.I. Ogbu and H.M. Samuelson (1977) "An Algorithm for the p-median Problem," Operations Research 25, 709-713.

G.L. Nemhauser and L.A. Wolsey (1978) "Best Algorithms for Approximating the Maximum of a Submodular Set Function," Mathematics of Operations Research 3, 177-188.

G.L. Nemhauser and L.A. Wolsey (1981) "Maximizing Submodular Set Functions: Formulations and Analysis of Algorithms," Annals of Discrete Mathematics 11, 279-301.

G.L. Nemhauser, L.A. Wolsey and M.L. Fisher (1978) "An Analysis of Approximations for Maximizing Submodular Set Functions I," Mathematical Programming 14, 265-294.

M. Padberg and S. Hong (1980) "On the Symmetric Travelling Salesman Problem: A Study," Mathematical Programming Study 12, 78-107.

B.T. Polyak (1969) "Minimization of Unsmooth Functionals," U.S.S.R. Computational Mathematics and Mathematical Physics, 14-29.

C.S. ReVelle and R.S. Swain (1970) "Central Facilities Location," Geographical Analysis 2, 30-42.

L. Schrage (1975) "Implicit Representation of Variable Upper Bounds in Linear Programming," Mathematical Programming Study 4, 118-132.

K. Spielberg (1969a) "Plant Location with Generalized Search Origin," Management Science 16, 165-178.

K. Spielberg (1969b) "Algorithms for the Simple Plant Location Problem with Some Side Conditions," Operations Research 17, 85-111.

T.J. Van Roy and L.A. Wolsey (1983) "Valid Inequalities for Mixed 0-1 Programs," CORE Discussion Paper 8316, University of Louvain, Belgium.